LA BIBLE MICRO APPLICATION

Laurent Guédon - Damien Heute
Thomas Heute - Pierre-Emmanuel MULLER

PHP 5

Micro Application

1ère Édition - Janvier 2005

Auteurs Laurent GUÉDON, Damien HEUTE, Thomas HEUTE et Pierre-Emmanuel MULLER

Avertissement aux utilisateurs

Les informations contenues dans cet ouvrage sont données à titre indicatif et n'ont aucun caractère exhaustif voire certain. A titre d'exemple non limitatif, cet ouvrage peut vous proposer une ou plusieurs adresses de sites Web qui ne seront plus d'actualité ou dont le contenu aura changé au moment où vous en prendrez connaissance.

Aussi, ces informations ne sauraient engager la responsabilité de l'Editeur. La société MICRO APPLICATION ne pourra être tenue responsable de toute omission, erreur ou lacune qui aurait pu se glisser dans ce produit ainsi que des conséquences, quelles qu'elles soient, qui résulteraient des informations et indications fournies ainsi que de leur utilisation.

ISBN : 2-7429-3608-4

Couverture réalisée par Stefanos ATHANASSOPOULOS

MICRO APPLICATION
20-22, rue des Petits-Hôtels
75010 PARIS
Tél. : 01 53 34 20 20 - Fax : 01 53 34 20 00
http://www.microapp.com

Support technique
disponible sur
www.microapp.com

Avant-propos

La collection *Bible Micro Application* a été conçue pour permettre aux utilisateurs avancés à experts d'approfondir leurs connaissances d'un thème précis. Exhaustifs, ces ouvrages permettent d'acquérir une connaissance intégrale du sujet étudié, à la fois en théorie et en pratique.

Conventions typographiques

Afin de faciliter la compréhension des techniques décrites, nous avons adopté les conventions typographiques suivantes :

- **gras** : menu, commande, boîte de dialogue, bouton, onglet.
- *italique* : zone de texte, liste déroulante, case à cocher, bouton radio.
- `Police bâton` : instruction, listing, texte à saisir.

✂ : indique un retour à la ligne dû aux contraintes de la mise en page.

Au cours de votre lecture, vous rencontrerez les encadrés suivants :

Propose des trucs pratiques.

Met l'accent sur un point important qu'il ne faut négliger à aucun prix.

Mentionne un fichier exemple ou un programme disponible sur le CD d'accompagnement de l'ouvrage.

Vous recommande une technique ou une marche à suivre.

Vous indique le nom et l'emplacement des fichiers à télécharger.

INTERNET

Renvoi à un site où vous trouverez des infos complémentaires ou des outils à télécharger.

REMARQUE

Il s'agit d'informations supplémentaires relatives au sujet traité.

RENVOI

Fait référence à un chapitre où vous trouverez des informations complémentaires.

Contenu en un clin d'œil

Sommaire

Laurent GUEDON

Je dédie ce livre à tous mes amis et plus particulièrement...

- À Philippe, Amiel (les compétences) et l'équipe de Felix ainsi qu'à la société Cyberbrain (mon couffin) ;
- À Myriam, si adorable ;
- À mes parents pour leur soutien ;
- À la winwin team, Zou et Syl.

Damien HEUTE

Parce qu'il y a des choses plus faciles à écrire qu'à dire, je dédie ce livre :

- À mes parents que j'aime et qui m'ont toujours apporté un soutien indéfectible ;
- À mes frères et à mon neveu qui peut-être, un jour, s'intéressera à la version 2015 de PHP ;
- À mes collègues qui par leur bonne humeur leur et professionnalisme m'ont beaucoup appris durant ma première expérience professionnelle ;
- À mes amis que j'ai un peu délaissés pour écrire ce livre.

Thomas HEUTE

Je dédie l'énergie apportée dans ce livre :

- À mes parents et frères que j'ai peu vus durant la dernière année ;
- Mon filleul Adrien, star du chapitre sur GD, et qui a encore le temps avant d'apprendre le PHP ;
- Mes amis de France et des États-Unis, que j'ai quelque peu délaissés (I'd like to thank my US friends and to apologize about having been so busy. By the way, we'll keep in touch) ;
- Aux membres actifs de l'ESIAL Roller Dream Team 1999-2000.

Pierre-Emmanuel MULLER

Merci à Amiel, Sylviane, Pierre, Régis Priqueler (http://www.aizenko.com) et Steph' Pineau (http://steph.pineau.free.fr/php/).

Préface

À qui s'adresse ce livre ?

Ce livre a été écrit avec, en constante toile de fond, les attentes des lecteurs à l'égard de ce que doit être la Bible d'un langage de programmation ; à savoir qu'elle doit pouvoir répondre aux interrogations du novice comme à celles de l'utilisateur expérimenté.

Une brève histoire du PHP, ainsi qu'une introduction à la philosophie du logiciel libre, vous montreront qu'autant PHP aide au développement de l'Internet autant Internet contribue à l'évolution de PHP et offre quantité de ressources qui lui sont liées. Les débutants ne seront pas laissés sur le bord du chemin dès les premières pages puisque les différentes possibilités d'installation de PHP font l'objet d'un chapitre complet et détaillé où les principaux éditeurs du marché sont également passés en revue.

De très nombreux aspects de PHP sont couverts dans cet ouvrage et la majeure partie des fonctions présentées sont illustrées par des exemples mettant bien en avant leur utilisation et comportement. Voilà pourquoi ce livre est plus qu'une simple documentation de base. Le PHP est replacé dans une optique d'apprentissage, ainsi que dans une idée d'utilisation au quotidien. Le lecteur apprendra vite à produire des formulaires, à créer des images, à manipuler des fichiers ainsi que des bases de données, ou encore à envoyer des e-mails. Le novice sera pris par la main pour qu'aucune zone d'ombre ne puisse subsister.

Même si la présentation de la programmation procédurale n'est pas délaissée, ce qui peut vous permettre de tirer partie de ce livre si vous restez attachés à la version 4 de PHP, la programmation orientée objet est bien évidemment présentée minutieusement afin de tirer pleinement partie de PHP5.

Une fois acquis les premiers rudiments, l'utilisateur devenu initié, trouvera dans ce livre de quoi aller plus avant dans sa maîtrise du PHP. Il pourra apprendre à utiliser des bases de données professionnelles ainsi que les annuaires LDAP. Il pourra manipuler les documents XML, créer des animations flash ou des fichiers PDF, accéder aux fonctionnalités réseau dont SOAP voire utiliser PHP en conjonction avec COM. Enfin, pour une utilisation véritablement professionnelle, il aura tout intérêt à consulter les comparatifs de solutions d'optimisation et d'obfuscation du code.

Comme le démontrent les témoignages d'utilisateurs qui ont fait le choix du langage PHP, bien qu'historiquement créé pour les besoins d'une page "perso", PHP a désormais sa place dans le monde de l'entreprise.

Les auteurs

Les quatre auteurs de cette Bible disposent tous d'une expérience différente de PHP. Venant d'horizons divers, ils ont tenu à mettre en commun leurs approches pour balayer le maximum d'aspects relatifs au PHP.

Damien HEUTE

Né en 1972, ingénieur E.N.S.P.S. (École nationale supérieure de physique de Strasbourg), spécialité traitement d'images, Damien a travaillé trois ans et demi en tant qu'ingénieur étude et développement pour un grand groupe européen d'aéronautique et de défense. Il y a mis en œuvre les technos C, C++/CORBA, Java, et, dans une moindre mesure, des technologies relatives à l'Internet et aux bases de données professionnelles. Il a ensuite passé dix mois en tant qu'ingénieur étude et développement pour une start-up spécialisée dans l'aide à la navigation via Internet sur des supports "sans fil", où il a mis en œuvre des technologies Java et JSP/Servlet. Maintenant dans un grand groupe français acteur majeur de la téléphonie mobile, de la biométrie et de la défense, il conçoit et développe, dans le cadre de gros projets, des applications web basées sur le langage Java.

Il pratique le PHP à titre personnel, en particulier pour les sites **http://www.ootoogo.com** et **http://www.toutestfacile.com**

Il aime le VTT et le cyclisme sur route, il fait un peu de photo et beaucoup d'informatique.

Thomas HEUTE

Né en 1977, il a suivi ses études à l'ESIAL (École supérieure d'informatique et applications de Lorraine) et possède de nombreuses compétences dans le domaine informatique. Thomas connaît bien les systèmes Windows, Linux ou UNIX. Les langages dans lesquels il est le plus à l'aise sont Java et PHP. Il maîtrise également de nombreuses bases de données, ainsi que des outils de conception de logiciels (UML ou MERISE).

Il a travaillé trois ans au National Institute of Standards and Technology (USA), où il a effectué des recherches sur la sécurité et les PDA après avoir travaillé sur les systèmes d'enregistrement d'objets par XML (ebXML).

Il est également à l'origine du site **http://www.toutestfacile.com**, site à vocation d'enseignement du PHP, du SQL et du XML.

Il s'investit dans le développement Open Source avec "Nukes the JBoss Portal" en Java/J2ee.

Il aime la batterie, la guitare et la photographie mais surtout l'informatique.

Laurent GUEDON

Né en mai 1975, ce qui, pour lui, est déjà très (trop) loin.

Après une formation E.E.A (électronique), Laurent, vouant une grande passion au développement Internet, a rejoint en 1999 une SSII spécialisée dans le développement d'applications et de sites Internet. Laurent est développeur PHP depuis PHP 2, ses premiers pas ayant été effectués sur l'ex-hébergeur AlternB. Développeur indépendant depuis septembre 2002, il est chargé d'enseignement à la faculté de Rennes.

Il pratique (plutôt mal) la guitare, fait du roller et se fait de petits trous dans la peau avec des piques en métal (il appelle ça le piercing). Mais, surtout, il ne se sépare jamais de son Psion qui lui permet de développer pendant ses déplacements.

Pierre-Emmanuel Muller

Journaliste spécialisé dans l'internet et les nouvelles technologies, il a un intérêt marqué pour tout ce qui touche à la protection de la vie privée ainsi qu'aux logiciels libres. Il a participé à diverses traductions d'applications libres. Il est l'auteur ou coauteur de plusieurs ouvrages de vulgarisation informatique (*Sécurisez votre PC*, *Montez votre serveur de A à Z*, *Utiliser les logiciels libres*, etc.).

Chapitre 1

Introduction

1.1. À quoi sert PHP ?

Présentation, rôle et fonctionnement d'un langage de script

Avant de présenter le langage PHP, il est utile de situer brièvement sa place sur l'internet. Comment, en effet, bien comprendre l'utilité d'un langage de script si la notion de "serveur" est encore floue ?

L'Internet est un réseau de réseaux, constitué d'ordinateurs connectés entre eux. Sa structure est maillée, non pyramidale. Il y a bien des échelons, mais les échelons les plus élevés sont, en fait, assez proches de la base, et sont, surtout, en très grand nombre. Ces échelons sont interchangeables. Quand votre échelon supérieur est absent, en panne, ou disparu, vous pouvez utiliser un autre échelon de même type. En bref, deux machines sont généralement interchangeables.

Il est ainsi très difficile de "diriger" l'Internet. Comme tout bon processus de communication, Internet dispose d'un langage (en fait de plusieurs langages) qui dépend du service que vous désirez utiliser (web, mail, ftp, etc.). Quel que soit le service en question, ce langage est commun à toutes les machines informatiques, et est indifférent aux plateformes. Les données provenant d'Internet sont donc, moyennant parfois quelques retraitements, réutilisables par tout ordinateur. Ainsi, un site web est construit indépendamment de la plateforme sur laquelle il est conçu. Ce sont les logiciels qui s'adaptent aux machines. Il existe un navigateur web (Netscape) pour PC et une autre version pour Mac. Les données qu'ils exploitent, en revanche, sont indépendantes des systèmes d'exploitation (Windows, MacOs, Linux, etc.). Les composantes les plus connues de l'Internet sont le Web (World Wide Web) ou le courrier électronique (e-mail).

Concentrons-nous sur le Web, car c'est de cela qu'il est question avec PHP. Admettons que vous désirez consulter le site web du moteur de recherche Google. Vous donnez l'adresse **http://www.google.com** à votre navigateur, vous lancez la recherche, et, quelques secondes plus tard, la page d'accueil du site s'affiche sur votre écran. En fait, derrière cette action se cache toute une interaction client-serveur, primordiale à intégrer pour bien saisir le fonctionnement de PHP. Votre navigateur, le client, interroge un serveur, en lui demandant la page d'accueil d'un site. Le serveur est une machine qui héberge le site web en question. Cette machine doit être, en principe, connectée en permanence au réseau. Concrètement, ce sont les disques durs de cette machine qui stockent les pages HTML ou PHP qui seront demandées par les internautes. Pour que le serveur soit en mesure de vous répondre, il faut que certains programmes et services soient présents et fonctionnels en son sein. Dans le cas d'un site web, il faut que le serveur dispose d'un serveur web, c'est-à-dire d'un programme permettant l'interprétation et la diffusion des pages demandées par les internautes. Votre navigateur demande la page, le serveur web reçoit et comprend votre requête, il recherche la page en question et, si elle est disponible, vous la retourne. En fait, il ne vous retourne que le code source de la page, en général du HTML, mais ce peut être une image ou une animation. La mise en forme, la construction et l'affichage de la page incombent à votre navigateur.

PHP est un langage de script HTML, c'est-à-dire qu'il fonctionne en relation avec le langage HTML (HyperText Markup Language). Il fonctionne du côté du serveur, et non pas du côté du client, et permet de générer des pages web "à la volée". Concrètement, un script PHP est intégré au code source d'une page HTML. Lorsque la page est appelée, le script PHP est interprété, et le tout est livré au serveur web qui, au final, répond au navigateur de l'internaute. Le surfeur obtient une page HTML tout ce qu'il y a de plus classique, à la différence que cette page

n'existait pas en tant que telle sur le serveur web. On dit alors que la page a été créée "à la volée", ou dynamiquement.

Plus concrètement, prenons une page HTML toute simple dont voici le code source :

```
<html>
  <head>
    <title>Je teste le langage PHP</title>
  </head>
  <body>
  </body>
</html>
```

Voici un script PHP sans prétention :

```
<?php
echo ("PHP est un palindrome.");
?>
```

Pour intégrer ce script à la page HTML, il suffit, ici, d'insérer le code entre les balises `<body>` et `</body>` de la page. Ce qui nous donne alors ceci :

```
<html>
  <head>
    <title>Je teste le langage PHP</title>
  </head>
  <body>
<?php
echo "PHP est un palindrome.";
?>
  </body>
</html>
```

C'est aussi simple que ça. Comme vous pouvez le constater, le script PHP est encadré par une balise de début et une de fin (à savoir `<?php` et `?>`). Ce sont des informations qui, à la lecture de la page, indiquent au serveur qu'il doit interpréter ces informations comme de PHP, et non plus comme du HTML.

Un script est défini comme suit par le jargon Linux France : "Une suite d'instructions simples, peu structurées, permettant d'automatiser certaines tâches [...]". Dit ainsi, c'est assez cinglant. En clair, un script PHP vous permet non seulement de générer du texte ou du code HTML, mais également d'envoyer un courriel, d'accéder à une base de données, de lancer un programme sur le serveur, etc. Il est donc ainsi possible de compter le nombre de visiteurs sur votre page, de gérer un livre d'or ou bien même de réaliser des applications web à part entière, comme des gestionnaires de contenus (PHPNuke, DaCode, SPIP, etc.) ou des outils de travail collaboratifs (PHPGroupware, par exemple).

En fait, PHP possède les mêmes fonctionnalités que les autres langages de script. Son gros atout réside dans son fonctionnement résolument tourné vers le Web. En natif, ce sont plus de vingt bases de données qui sont supportées par PHP, sans que l'utilisateur ait à faire quoi que ce soit. Tous les grands protocoles de l'internet fonctionnent, eux aussi, sans autre forme de procès avec PHP : SMTP, POP3, IMAP, FTP, SNMP, etc.

1.2. Les version 1 à 5 de PHP

Rasmus Lerdorf

Chaque grand projet libre a un père ou un leader charismatique. PHP ne déroge pas à la règle. Qui plus est, l'histoire de PHP facilite les choses, ses premiers pas étant tout ce qu'il y a de plus simple à narrer. Les premières esquisses de PHP sont le fait d'un seul homme : Rasmus Lerdorf. Sur son site personnel, Rasmus Lerdorf reste assez discret. Les rares éléments que l'on peut glaner sur son parcours personnel se limitent à des expériences professionnelles ou à quelques hauts faits de programmation. On en apprend même plus en suivant l'histoire de PHP, intimement liée à celle de son créateur, tout au moins dans les premières années. Rasmus Lerdorf est né en 1969 sur l'île de Greenland. On comprend que le petit Rasmus ait préféré l'informatique à l'agriculture, tant l'île de Greenland est peu clémente : les terres arables, les forêts et les pâturages permanents représentent à peine 1 % de la superficie totale de l'île. L'île de Greenland est tout de même d'une superficie trois fois supérieure à celle du Texas (2 170 000 km^2) pour une population inférieure à celle de l'île de Guernesey qui ne fait que 78 km^2 (56 300 personnes).

Figure 1.1 : *Rasmus Lerdorf*

Rasmus Lerdorf ne se définit pas comme un programmeur, même si le fait qu'il soit capable de faire fonctionner PHP sur son magnétoscope conduit le spectateur à penser le contraire. Il n'a pas de diplôme d'informaticien, mais une formation d'ingénieur. Il a travaillé pour des sociétés comme Bell, IBM ou Linux Care. Selon une étude menée à grands coups de scripts Perl par Per Abrahamsen, Rasmus Lerdorf est l'un des Danois les plus célèbres. Il arrive même, dans le classement, avant le penseur Soren Kierkegaard.

Signalons enfin que Rasmus Lerdorf, en bon développeur du logiciel libre, n'est pas dénué de sens de l'humour. Le petit dernier des Lerdorf a pour nom Carl Alexander Lerdorf. Sur le site de la famille Lerdorf, son nom se lit : Carl AlexandeR Lerdorf… Récursif, quand tu nous tiens !

INTERNET

Quelques sites (en anglais)
Site officiel de la famille Lerdorf (avec les photos du petit Carl Alexander Lerdorf) :
http://www.lerdorf.com
Les Danois célèbres :
http://www.dina.kvl.dk/~abraham/fame/fame.html

Naissance et évolution de PHP

Naissance et première version

La première version de PHP n'en était pas une. Ce ne furent que quelques outils développés par Rasmus Lerdorf pour les besoins de son site personnel. Dans le courant de l'année 1993, M. Lerdorf avait commencé à développer des scripts en langage C et en Perl. À partir de là a pu être constituée, en septembre 1993, une première librairie. Lassé de devoir réécrire encore et toujours les mêmes portions de code, M. Lerdorf eut l'idée de séparer sa logique de programmation de celle du HTML, afin de pouvoir réutiliser plus facilement certaines portions de code.

Pour qu'une première version de PHP comme langage de script voie le jour, il ne manquait plus à la librairie qu'un interpréteur capable d'analyser le code HTML, pour y repérer des balises particulières et appeler les fonctions qui y sont liées. Cela fut fait en novembre 1993, date à laquelle M. Lerdorf situe la naissance de la toute première version de PHP. La première mouture du langage n'avait donc pas de très grandes ambitions. PHP se limite alors à un interpréteur qui analyse une page HTML, en ressort des balises (inspirées du SGML), et appelle les fonctions correspondantes à ces balises. Tout cela dans quel but ? Afin de compter et d'enregistrer le nombre de visiteurs consultant un curriculum vitae sur son site personnel. Rasmus Lerdorf résume ses motivations à une lassitude face à Perl, jugé comme étant trop lent et doté d'un analyseur trop restrictif.

Cette première version n'eut qu'un succès très limité, et pour cause… elle ne fut jamais publiée. Il faut attendre février 1994 pour qu'une première version de PHP soit enfin diffusée.

INTERNET

Interviews de Rasmus Lerdorf
Interview de Rasmus Lerdorf dans le Journal du Net :
http://developpeur.journaldunet.com/itws/it_phpnexen_rasdorf.shtml
Interview de Rasmus Lerdorf pour O'Reilly (en anglais) :
http://web.oreilly.com/news/lerdorf_0200.html

PHP 2 et l'ouverture sur le monde

La première utilisation plus large de PHP remonte au site http://www.io.org. Rasmus Lerdorf réutilisa son langage de script pour installer divers petits outils en ligne (un compteur de visiteurs, l'affichage de la dernière personne connectée, etc.). Alors, le succès de PHP fut immédiat. D'autres utilisateurs du site voulurent utiliser PHP, et les scripts poussèrent dans le HTML comme les champignons dans un sous-bois humide. Étant donné que l'interpréteur évoqué plus haut était lui-même un CGI, les administrateurs du site repérèrent rapidement le changement. PHP se révélait fort gourmand en ressources. Pour remédier à cela et éviter une mort certaine du langage, Rasmus Lerdorf se pencha alors sur le serveur web pour y inclure PHP comme module interne plutôt que de le conserver en module externe (CGI), beaucoup plus lourd. Le serveur utilisé était HTTPD NCSA, de l'Université de l'Illinois. Une fois PHP ajouté au serveur, M. Lerdorf convainquit tout de même les administrateurs de passer sous Apache, serveur web beaucoup plus souple à utiliser et à programmer.

C'est à cette époque que M. Lerdorf obtint un emploi à l'université de Toronto. Sa mission était de mettre en place un système de connexion à l'Internet géré depuis une interface web. Le tout faisait intervenir une base de données des étudiants, des serveurs, des terminaux Cisco ainsi que quelques autres composants. En recherchant un outil utilisable dans la mise en place de cette architecture, Rasmus Lerdorf se rendit compte qu'il n'y avait rien qui lui plaisait réellement. Il décida donc de réécrire l'interpréteur de PHP et ses outils connexes, pour les rendre plus facilement multi-usages et généralistes. PHP devait être capable de dialoguer avec des bases de données ou d'autres sources extérieures (comme, par exemple, des formulaires HTML).

Ce sera à nouveau, pour Rasmus Lerdorf, l'occasion d'utiliser PHP et de le faire progresser. Anciennement Personal Home Page, PHP s'appelle alors PHP/FI (PHP/Form Interpreter, interpréteur de formulaires), du fait des nouveaux outils ajoutés par M. Lerdorf pour interfacer PHP avec des bases de données. La syntaxe de PHP/FI reste simple, encore un peu incohérente et assez proche du Perl.

PHP/FI connaît alors un très large succès. De nombreux développeurs utilisent PHP, et non pas seulement pour interroger des bases de données. PHP/FI sert déjà à créer des pages web au contenu dynamique, ce qui sera l'une des applications les plus populaires du langage par la suite. Il présente déjà certaines des principales fonctionnalités de PHP.

En 1997, ce sont déjà plus de 50 000 sites qui utilisent PHP.

PHP 3

Publiée en juin 1998, après neuf mois de tests, la troisième version du désormais officiellement nommé PHP (PHP : Hypertext Preprocessor, le premier P faisant référence à PHP lui-même. Récursif quand tu nous tiens...) fait un grand bond en avant. La sortie de cette nouvelle version correspond également à une solide refonte du moteur de PHP. Deux développeurs, Zeev Suraski et Andi Gutmans jugeaient que PHP/FI n'était pas assez performant pour leurs applications de commerce en ligne, et décidèrent donc de le réécrire. PHP 3 fut annoncé comme le successeur officiel de PHP/FI.

L'une des principales nouveautés de PHP 3 – principale en tout cas au regard de son développement – fut une API (Application Programming Interface, interface de programmation d'application) qui permit à de nombreux développeurs de participer au développement du langage. De plus, PHP supportait désormais la syntaxe objet, sa syntaxe propre devenant, elle, plus cohérente et solide.

La version finale de PHP 3 fonctionnait sur de nombreuses plateformes, avec de nombreux serveurs web et bases de données. Les protocoles SNMP ou IMAP étaient alors supportés moyennant une compilation appropriée. PHP pouvait être utilisé en mode CGI avec la plupart des serveurs HTTP, ou bien être chargé en module, comme par exemple avec Apache.

De nombreuses fonctionnalités avaient été ajoutées à PHP 3, mais, en plus, cette sortie coïncidait avec l'attente de nombreux nouveaux webmestres qui avaient besoin d'un outil simple pour ajouter une touche de dynamisme à leurs sites web. La fin des années 1990 fut, en effet, une période faste pour le Web, porté dans sa croissance par l'arrivée de l'Internet dans le grand public. PHP permettait alors de mettre en place très facilement une solution de commerce en ligne, des albums photo, des systèmes de gestion de contenus, etc. Sa facilité d'accès et d'usage, ainsi qu'une gamme d'outils résolument tournés vers le Web, faisaient de ce langage la solution idéale pour de nombreux webmestres.

À la fin de l'année 1998, plus d'un million de serveurs web devaient utiliser PHP 3. Le langage de script avait conquis plusieurs dizaines de milliers d'utilisateurs.

PHP 4

Fidèle à la toute jeune tradition, PHP 4 allait une nouvelle fois entraîner une réécriture complète de son moteur. Dès l'hiver 1998, Andi Gutmans et Zeev Suraski se penchèrent sur le moteur interne du langage pour le reprendre de fond en comble. Leur objectif était de rendre PHP encore plus performant (ce que l'on comprend aisément), et de rendre le code encore plus modulaire. Il fallait également permettre au PHP d'être plus à l'aise avec des applications complexes. PHP 3 posait problème face à ces applications complexes. Sa syntaxe permettait leur exécution, mais le moteur du langage n'avait pas été conçu pour les faire fonctionner efficacement. L'écriture d'un nouveau moteur semblait alors nécessaire pour permettre au PHP de passer à une nouvelle étape de sa croissance.

Ce nouveau moteur devait porter le nom de ses deux créateurs : le Zend Engine, combinaison de ZEev et ANDi. Une première version de PHP doté de ce nouveau moteur sortit dans le courant de l'année 1999. La publication officielle de la version finale fut annoncée en mai 2000, six ans après le premier PHP. PHP 4 assurait une compatibilité ascendante par rapport à PHP 3, et la migration de PHP 3 à PHP 4 fut encore plus souple que celle de la version 2 à la version 3.

Outre un cœur neuf, PHP disposait d'une pléthore de nouveaux membres. Dans sa quatrième version, PHP supporte quasiment tous les serveurs web du marché, les bases de données les plus répandues, plusieurs nouvelles structures de langage ou encore la bufferisation de sortie. On compte également des améliorations en matière de sécurité. L'architecture de PHP 4 est encore plus ouverte et évolutive que celle de PHP 3. On remarque notamment la création d'une interface ISAPI (Internet Server Application Programming Interface, outil de spécification des DLL pouvant être utilisé par les serveurs web de Microsoft). PHP 4 permet également d'instancier et de manipuler des classes Java comme si elles étaient de simples classes PHP. PHP peut aussi exécuter des servlets. Autre amélioration majeure de PHP 4 : l'introduction des sessions en mode natif. L'impossibilité pour PHP 3 de gérer les sessions constituait une grosse lacune pour un langage tout entier tourné vers le Web. Cette faiblesse ne jouait pas en faveur de PHP, tout particulièrement face à l'ASP qui, lui, savait déjà gérer les sessions. La lacune de PHP 3 pouvait certes être comblée via des solutions comme PHPLib, mais l'inclusion en mode natif dans PHP 4 réglait définitivement la question. PHP 4 marque également l'entrée du langage dans le monde des sociétés commerciales. Les deux pères fondateurs de PHP 3, Zeev Suraski et Andi Gutmans, créèrent la société Zend pour distribuer et vendre des produits gravitant autour de PHP. Si le Zend Engine reste libre, bien que sous une licence assez critiquée jusque dans sa seconde version, certains produits commercialisés par Zend sont des logiciels propriétaires. Signe évident d'une maturité du langage, PHP devient alors le support d'un commerce tourné vers les grands comptes de l'Internet. Belle évolution pour ce qui n'était, au départ, qu'une petite collection de scripts Perl destinés à compter des visiteurs...

La communauté PHP ne compte plus seulement des développeurs ou des webmestres. Désormais, de nombreux internautes participent à la documentation de PHP, à sa traduction ou à son annotation. En France, par exemple, la société Nexen héberge la documentation officielle de PHP en français, dont elle a produit une version annotée. Le projet Pear regroupe, lui aussi, de nombreux participants, qui ne sont pas directement impliqués dans le développement de PHP.

On compte aujourd'hui plusieurs millions de sites web qui utilisent PHP. En se basant sur ce comptage des noms de domaines, on peut dire que cela représente 20 % de l'Internet. Les développeurs, eux, sont des centaines de milliers à l'utiliser. Le site Security Space tient à jour une liste des modules Apache les plus utilisés. En juin 1998, PHP était présent sur 8 % des sites audités par Security Space. En mars 2002, presque 45 % des sites audités avaient installé le

module PHP pour Apache. Perl, lui, était, à la même date, présent sur 20 % de ces sites (rappelons que Perl date de 1987 et PHP de 1994).

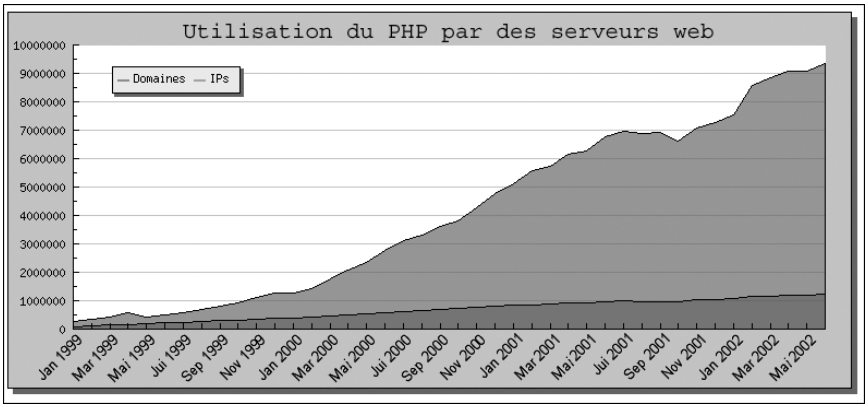


Figure 1.2 : *Le nombre de sites utilisant PHP croît de jour en jour*

PHP 5

La tant attendue version 5 de PHP est sortie le 13 juillet 2004. Elle apporte de très nombreuses nouveautés tout en conservant une large compatibilité avec PHP4. Si la grande nouveauté de la version 5 reste le nouveau modèle orienté objet, les développeurs de PHP n'ont pas oublié l'héritage des 4 précédentes versions qui ont permis à de très nombreux développeurs de faire leurs premiers pas dans le monde du script pour le web. La plupart des scripts écrits pour PHP4 devraient fonctionner sans modifications avec PHP5. Il existe toutefois quelques différences dont la liste est disponible sur le site officiel du projet PHP.

Incompatibilités entre PHP4 et PHP5
Même si vos scripts écrits pour PHP4 ont de très larges chances d'être compatibles avec PHP5, certaines incompatibilités ou modifications sont connues et listées. Le site officiel PHP en dénombre une dizaine. Vous les trouverez à cette adresse :
http://www.php.net/manual/fr/migration5.incompatible.php

L'évolution de PHP

PHP, et avec lui toute la communauté de développeurs, doit faire face à des questions inédites. Sa taille provoque des soucis d'échelle, déjà rencontrés par d'autres projets (comme Linux, par exemple). Jusqu'à présent, les réponses apportées par la communauté aux problèmes apparus furent satisfaisantes, tout au moins assurèrent-elles sa survie et sa croissance, sans nuire à la qualité du langage. On peut citer, à titre d'exemple, la mise en place d'une équipe d'assurance qualité durant l'été 2000 pour faire face aux nombreuses critiques suscitées par la piètre qualité des premières versions de PHP 3. Le langage avait, en effet, été testé sur trop peu de plates-formes différentes. Désormais, une équipe complète de développeurs a pour objectif principal de chasser le bug au sein du projet PHP.

Alors que PHP est un langage de script né du Web pour le Web, l'on voit même naître des projets impliquant PHP, qui n'ont plus rien à voir avec les objectifs premiers du langage. PHPGtk est ainsi une solution permettant de développer des applications en PHP à partir d'une interface graphique côté client.

De même, l'entrée en jeu de sociétés commerciales a amené des questionnements soulevés par des acteurs du monde du libre, comme la FSF (cf. problème de la licence du moteur Zend). Encore une fois, la communauté PHP a su faire preuve de beaucoup de pragmatisme dans ses réponses. Résultat des pressions de la FSF, Zend a changé la licence de son moteur. PHP 4 reste, pour sa part, sous une licence Apache, jugée plus souple pour les développeurs. Et, quand on demande aux développeurs de PHP comment ils prennent les critiques que l'on peut leur adresser parce que PHP 4 n'est pas GPL, ils vous répondent "avec des frites".

Si Rasmus Lerdorf a pu être, à un moment, une figure de proue de PHP, et qu'il reste aujourd'hui encore son représentant le plus médiatique, on peut tout de même remarquer que la communauté PHP ne bénéficie pas d'une structure centralisatrice qui ferait l'interface entre le langage et les entreprises ou les institutions. Le logiciel libre a l'Open Source Initiative ou la FSF, un projet comme Kde dispose de toute une infrastructure qui assure sa promotion ou sa représentation, Linux a Linus Torvalds. Vu de l'extérieur, le monde de PHP paraît encore flou.

INTERNET

PHP
La société Zend, éditrice du Zend Engine (en anglais) :
http://www.zend.com
La société Nexen :
http://www.nexen.net
Nombre de sites utilisant PHP (en anglais) :
http://www.php.net/usage.php
Modules Apache les plus utilisés (en anglais) :
http://www.securityspace.com/s_survey/data/man.200203/apachemods.html
Des projets attachés au PHP :
http://php3.de/manual/fr/html/history.php.related.html
Interview du créateur de PHPGtk (en anglais) :
http://beta.usephp.net/article.php3?id_article=3p

La création de la communauté

Si PHP a pu croître et prospérer jusqu'à devenir l'un des langages de script les plus utilisés, c'est en grande partie grâce à une communauté de développeurs qui, dès la version 3, fut la cheville ouvrière de ses améliorations. PHP/FI, seconde version du langage, reste l'œuvre d'un seul homme, Rasmus Lerdorf, père de PHP. À partir de la version 3, Rasmus Lerdorf n'est plus seul à la tête de PHP. Zeev Suraski et Andi Gutmans prirent en main, par deux fois, la refonte complète du moteur de PHP. Cette ouverture de PHP est due à une interface API mise en place très tôt, mais, surtout, à la volonté de son créateur de partager son travail avec d'autres développeurs. Rasmus Lerdorf le dit lui-même : il a toujours appris en regardant le code d'autres programmeurs. Ne pas faire profiter, en retour, la communauté de son travail ne lui a même pas traversé l'esprit.

Comment cette communauté est-elle née ? Dans une interview, Rasmus Lerdorf situe sa date de naissance au jour où, alors employé par l'université de Toronto, il avait mis en ligne son code source à disposition sur une page web. Cela dans l'optique évidente que d'autres puissent en profiter. Il n'avait alors aucune idée de qui utilisait PHP de par le monde, ni pour quelles applications. Un beau jour, il reçut un courriel d'un utilisateur japonais, qui lui faisait parvenir un patch. Le patch était tout à fait fonctionnel et pertinent. Il résolvait un problème auquel Rasmus Lerdorf n'avait pas encore eu affaire, mais qui allait se poser inéluctablement. PHP bénéficiait d'une aide extérieure, la communauté était née ; elle compte aujourd'hui plus de 300 développeurs.

La communauté PHP
Rasmus Lerdorf raconte la naissance de la communauté :
http://developpeur.journaldunet.com/itws/it_phpnexen_rasdorf.shtml

1.3. La communauté du libre

Pourquoi présenter le monde du logiciel libre dans un livre traitant de PHP ? Simplement parce que, sans cette communauté, PHP ne serait pas ce qu'il est aujourd'hui. Si la marque du monde du libre n'est pas particulièrement visible sur le langage lui-même, sa philosophie et son mode de développement ont eu, et ont toujours, une influence notable sur le langage PHP et sa croissance. Il suffit de voir le rôle prépondérant que joue le triptyque Apache-MySQL-PHP dans le monde du Web face à des solutions comme IIS-ASP-Access. PHP est indissociable du monde du libre. Cette partie n'a pas pour objectif de faire l'apologie du logiciel libre, ni même de faire un comparatif entre code source ouvert ou code source fermé ; tout comme le souligne Linus Torvalds lui-même, les deux existent et c'est tant mieux. Il nous a pourtant semblé utile de présenter le monde du libre, pour permettre à celui qui découvre PHP, d'une part de comprendre le contexte de sa croissance et de son évolution, et d'autre part de savoir vivre dans le monde de PHP. Celui qui ne sait pas que les développeurs qui travaillent avec PHP distribuent largement leurs sources et discutent massivement en ligne de l'évolution du langage et des questions techniques qui s'y rattachent sera vite perdu et passera à côté de l'un des principaux atouts de PHP : sa communauté.

Historique

Le monde du logiciel libre plonge ses racines dans l'histoire des hackers (au sens premier du terme). Toute la culture du logiciel libre et son folklore doivent beaucoup aux pionniers de l'informatique, ces "vrais programmeurs" : les hackers (selon la définition d'Eric S. Raymond). Pour une histoire complète et documentée de cette culture, on se reportera à *Une brève histoire des hackers*, par M. Eric S. Raymond. Disons simplement que les logiciels libres ne sont pas nés avec Linux. Leur histoire est bien plus riche et complexe.

C'est dans l'un des plus prestigieux laboratoires d'informatique du monde, celui d'intelligence artificielle du MIT (Institut de technologie du Massachusetts) que s'est cristallisée l'opposition entre logiciels libres et logiciels propriétaires. Chef de file du laboratoire, et féroce opposant à la commercialisation des techniques mises au point dans ce laboratoire, Richard M. Stallman créa, en 1984, la Fondation du logiciel libre (Free Software Foundation, FSF). Hacker au sens

propre du terme, il développa alors des logiciels libres, dont le célèbre éditeur Emacs. M. Stallman entrepris alors de recréer un clone complet libre du système d'exploitation propriétaire UNIX. C'est ainsi que naquit le projet GNU (GNU is not UNIX. Décidément, acronyme récursif quand tu nous tiens…). La FSF avait pour objectif de soutenir et d'accompagner des projets libres. Sa création formalisait les thèses et idées des défenseurs du logiciel libre. Fortement imprégnée de la culture de l'échange et de la transmission libre des savoirs des milieux universitaire et scientifique, la FSF estime que "tout comme les idées, les logiciels ne sont pas tangibles et peuvent être copiés sans perte. Les transmettre est la base d'un processus d'évolution qui alimente le développement des réflexions." (site de la FSF Europe, Qu'est-ce qu'un logiciel libre ?). Pour être qualifié de libre, un logiciel doit, selon la FSF, répondre à quatre critères de liberté. Tout d'abord, un programme doit pouvoir être exécuté pour n'importe quel usage. Ensuite, un utilisateur doit pouvoir étudier le fonctionnement du programme et l'adapter à ses besoins. Il doit également être possible de redistribuer des copies et, enfin, d'améliorer le programme en rendant publiques les modifications pour que l'ensemble de la communauté en bénéficie.

Pour assurer ces libertés aux logiciels libres, M. Stallman établit la GPL : GNU General Public License.

La machine en marche

Les logiciels de la FSF ne sont pas les seuls produits libres. De très nombreux logiciels libres sont en rapport avec l'internet. S'ils n'y sont pas directement liés, c'est leur développement qui en a profité.

On peut citer Sendmail, développé en 1981 par Eric Allman. Aujourd'hui encore distribué gratuitement par la société Sendmail Inc., Sendmail est l'agent de transport de mails le plus utilisé sur Internet. Il détient plus de 75 % des parts de marché.

Autre figure historique et emblématique du monde du libre : Perl. Perl signifie Pratical Extraction and Report Language (langage pratique d'extraction et de rapport). C'est un langage de programmation très largement utilisé sur l'Internet. Il sert principalement aux administrateurs de systèmes et de réseaux. Il sert également dans le développement de CGI. On estime qu'une centaine de programmeurs participe au développement de Perl. Les programmeurs utilisant Perl seraient 500 000 et les utilisateurs, eux, plusieurs millions. En se basant sur les ventes d'ouvrages dédiés à chaque langage, les éditions O'Reilly estiment que Perl est aussi souvent utilisé que Java.

Apache est un projet libre d'envergure ; peut-être celui qui a démontré le premier qu'un logiciel libre pouvait faire jeu égal avec les grandes entreprises de développement de logiciels. Il détient aujourd'hui la plus importante part de marché pour les serveurs web (source Netcraft), et cela face à des acteurs aussi puissants que Microsoft ou Netscape. En juin 1998, le géant de l'informatique IBM annonçait soutenir officiellement le groupe Apache. Ce choix d'Apache par IBM fut une étape importante dans l'histoire des logiciels libres. Les impératifs commerciaux et économiques qui président aux décisions d'IBM donnaient à ce soutien à Apache une valeur toute particulière. Un logiciel libre pouvait servir les intérêts du monde commercial sans se galvauder ni perdre ses ambitions.

L'histoire de la FSF n'est pas jonchée seulement de réussites. L'un des principaux échecs de la FSF fut de mettre beaucoup trop de temps pour fournir un noyau à son système d'exploitation (GNU). Cependant, l'une de ses principales réussites fut de fournir une trousse à outils à celui qui allait développer le noyau libre le plus connu : Linus Torvalds. C'est en effet grâce aux

logiciels de la FSF que le jeune étudiant de l'université d'Helsinki a pu développer son propre UNIX libre, Linux. Linus Torvalds voulait développer un UNIX pour son ordinateur personnel. Après avoir établi les premiers éléments de son système d'exploitation, il mit rapidement le code source à la disposition des internautes du monde entier. C'est là que réside l'une des particularités les plus importantes de Linux. Il est développé et maintenu par des programmeurs du monde entier, grâce à l'Internet. De grands acteurs de l'industrie informatique font désormais confiance à Linux (IBM, Sun, Dell, HP, etc.).

Cathédrale et bazar

Ayant observé le développement de Linux, Eric S. Raymond a voulu théoriser ce nouveau modèle de développement. Schématiquement, il oppose deux modèles : la cathédrale et le bazar. Le modèle en cathédrale est un modèle classique. C'est celui des entreprises et administrations classiques : centralisé, très hiérarchisé, il met beaucoup de temps à réagir, et les décisions sont soumises à de multiples validations. Dans le modèle bazar, tout est beaucoup plus horizontal, mouvant et flexible. Un modèle ouvert et collaboratif comme le bazar permet une évolution rapide et pertinente. Chacun est libre de soumettre un correctif ou une amélioration, ce qui évite de brider les initiatives.

La scission : l'Open Source

Richard Stallman n'a jamais fait l'unanimité dans le monde du logiciel libre. Son dogmatisme et ses prises de position fermes étaient vues par beaucoup comme des freins au développement plus rapide du monde du libre, tout particulièrement en direction des entreprises. Même si la licence GPL n'interdit pas du tout la commercialisation d'un logiciel libre, la FSF véhiculait une image hostile au monde du commerce. D'aucuns pensaient que les idées développées par la FSF nuiraient plus à la croissance de la communauté du libre qu'elles ne la serviraient. De ce fait, des programmeurs et d'autres acteurs du logiciel libre décidèrent de substituer le terme Open Source à celui de Free Software, le terme "free" étant jugé trop ambigu (en anglais, il désigne aussi bien ce qui est gratuit que ce qui est libre). Afin de ne plus effrayer le monde de l'entreprise, le mouvement Open Source vit donc le jour. On retrouvait à sa tête certains des plus prestigieux acteurs du monde du libre comme, par exemple, Bruce Perens (ancien mainteneur du système d'exploitation Debian http://www.debian.org et fondateur du projet de standardisation Linux Standard Base) ou Eric S. Raymond. L'objectif de ce nouveau mouvement était clairement exprimé : il s'agissait de mieux vendre et promouvoir le logiciel libre.

Il est important de bien comprendre que le courant de l'Open Source et celui des logiciels libres ne sont pas antinomiques, mais, qu'au contraire, ils sont complémentaires. Ils font la promotion du logiciel libre dans différentes directions. Toutefois, la FSF voit en l'Open Source un danger potentiel pour les logiciels libres. En effet, les points de divergence ne se situent pas seulement dans les termes, mais également dans les licences, la définition du logiciel libre et l'utilisation qui peut en être faite.

Pour qu'un programme soit dit open source, il faut qu'il respecte un certain nombre de critères définis comme suit par le mouvement Open Source. Tout d'abord, le logiciel open source doit pouvoir être librement donné ou vendu, sans que cela entraîne l'acquittement d'une redevance ou de droits d'auteur, et ce pour tous les utilisateurs de ce logiciel, sans distinctions. Le logiciel doit être distribué avec son code source, qui peut être modifié ou réutilisé par l'utilisateur à condition que ce dernier respecte la licence du programme originel. Un logiciel peut être open source même si sa licence n'autorise pas explicitement la réutilisation de son code source. À ce

moment là, la diffusion de patches (fichiers de modification) doit être autorisée. Une licence open source peut également demander à ce qu'une version modifiée d'un logiciel n'ait pas le même nom que le logiciel "modèle". Elle ne doit pas non plus établir de discrimination à l'encontre de personnes ou de groupes de personnes, ni même de distinctions relatives aux domaines d'utilisation du logiciel (il peut être commercial ou exploité dans des domaines qui sont sujets à débat, comme la recherche génétique par exemple). La licence ne doit pas être limitée à un ensemble de logiciels. Chaque logiciel formant une "suite" doit bénéficier de la licence open source. C'est la neuvième clause de la définition officielle de l'Open Source qui différencie principalement le mouvement Open Source de celui du logiciel libre. Elle dit : "La licence ne doit pas imposer de restrictions à d'autres logiciels distribués avec le programme. Par exemple, la licence ne doit pas exiger que tous les programmes distribués sur le même support soient des logiciels open source." Malgré cela, l'Open Source reconnaît la GPL comme étant compatible avec ses définitions : "La licence GPL est bien conforme à cette clause. Les bibliothèques sous licence GPL "contaminent" seulement les logiciels avec lesquels elles sont liées statiquement lors de la compilation, pas les logiciels avec lesquels elles sont distribuées" (justification de la clause 9 dans la définition de l'Open Source). Même si la porte reste ouverte, la FSF, elle, ne se reconnaît pas dans l'Open Source. Elle continue d'affirmer son attachement à la portée philosophique du logiciel libre et considère l'Open Source comme "quelque chose de proche (mais pas d'identique) au "logiciel libre"(cf. site de la FSF).

INTERNET

Quelques liens
La définition de l'Open Source :
http://www.idealx.org/fr/doc/fr-osd/fr-osd.html#toc1
La définition du logiciel libre :
http://www.fsfeurope.org/documents/freesoftware.fr.html
Licences compatibles avec la définition de l'Open Source :
http://www.idealx.org/fr/doc/fr-licences/fr-licences.html
Licences compatibles avec la définition de la FSF :
http://www.gnu.org/licenses/license-list.fr.html

Le droit et les logiciels libres

Signe de sa richesse et de sa complexité, le monde du logiciel libre regorge de licences sous lesquelles un développeur peut placer son travail. On compte plus d'une dizaine de licences dites "libres" ou apparentées. Pourquoi un tel foisonnement ? Parce que le logiciel libre ne date pas d'hier… et que les acteurs sont nombreux. Rien n'empêche une entreprise ou un particulier de créer sa propre licence se voulant libre. On peut citer les cas de Sun, Netscape, ou Apple, qui créèrent leurs propres licences.

Parmi toutes ces licences, on peut tout de même noter que les plus répandues sont : la GPL et la LGPL, la licence FreeBSD ou la MPL.

PHP est un langage de programmation libre et ouvert. Il propose, par exemple, une API (Application Programming Interface, une interface de programmation d'applications) qui permet à des développeurs d'ajouter des fonctions au langage.

Mais, encore une fois, rien n'est simple. Les différentes versions de PHP n'ont pas été diffusées sous la même licence. PHP 3 a été diffusé sous licence GPL, tandis que PHP 4 n'est pas

compatible avec la GPL. La licence actuelle (2.02) fait de PHP 4 un logiciel libre, mais il n'est pas reconnu comme compatible avec la GPL. La FSF encourage l'utilisation de PHP 3, diffusé sous GPL. Les responsables de PHP expliquent leur choix en raison des aspects trop contraignants de la GPL. Les développeurs ont décidé de publier PHP 4 sous une licence plus souple, pour permettre au PHP d'être le plus populaire possible. La licence de PHP 4 se rapproche de la licence d'Apache (PHP fait partie de la fondation Apache). Autre pomme de discorde : le Zend Engine, distribué avec PHP 4. Jusqu'en novembre 2001, le Zend Engine était publié sous une licence QPL, non compatible avec la GPL. Depuis, Zend a changé la licence de son moteur pour une licence de type BSD, compatible, elle, avec la GPL.

En tout état de cause, avant de faire quoi que ce soit avec PHP, lisez attentivement sa licence et celle de tout produit connexe que vous utiliseriez, pour savoir s'ils sont compatibles avec ce que vous voulez en faire.

Le texte (en anglais) des licences
Licence PHP 4 :
http://www.php.net/license/2_02.txt
Licence Zend Engine :
http://www.zend.com/license/2_00.txt

PHP et le libre

Et la communauté PHP dans tout ça ? Comme tout projet libre, PHP bénéficie de nombreuses contributions apportées par des développeurs bénévoles impliqués partout dans le monde. On estime à plus de 300 membres la communauté des développeurs PHP. Projet collaboratif, il tire avantage de tous les aspects positifs d'un développement libre. Les bugs découverts peuvent notamment être corrigés rapidement.

C'est grâce à son ouverture que PHP a pu croître aussi vite. De par son API, il s'est enrichi de nombreuses fonctionnalités comme, par exemple, le support des protocoles les plus utilisés sur l'internet (comme POP3, IMAP, ou NNTP), un accès aux annuaires LDAP, ou bien encore un parseur XML.

Langage né pour le Web, PHP dispose d'une communauté en ligne très active. Le père de PHP, Rasmus Lerdorf, le dit lui-même : "PHP a été développé par la communauté, et non pas par moi" (interview à Nexen pour le Journal du Net). D'une certaine manière, on peut comparer le développement de PHP à celui de Linux. Rasmus Lerdorf a eu le même rôle que Linus Torvalds. Il a initié le mouvement, a posé les premières bases, puis a permis à des internautes d'utiliser sa création et d'en connaître le code source. Ces étapes préalables réunies, la création spontanée d'une communauté de développeurs n'était plus très loin.

De nombreux sites proposent des centaines de scripts au téléchargement. Les développeurs qui utilisent PHP n'hésitent pas à faire profiter le reste de la communauté de leurs créations. Que ce soit sur des sites "perso" ou des sites connus, un programmeur qui veut mettre en place un forum, un site de gestion de contenus, ou même un freemail (de type Hotmail ou Caramail) ou un système de *knowledge managment* trouvera, à coup sûr, des programmes en PHP qui répondent déjà à ses attentes. L'un des premiers réflexes d'un programmeur en PHP qui travaille sur un nouveau projet sera d'aller chercher en ligne ce qui a déjà été fait dans son domaine, pour s'en inspirer ou pour adapter une solution. Le projet PEAR (PHP Extension

and Add-on Repository, ou dépôt d'extension et d'add-on PHP – *pear* signifie poire en anglais) représente bien l'esprit open source et "opportuniste" de PHP. Largement inspiré du système CPAN (Comprehensive Perl Archive Network) pour le langage Perl, Pear est une base de données en ligne de programmes ou d'extensions en PHP. Les développeurs partagent ainsi leurs créations. PHP a su récupérer l'un des aspects les plus intéressants du monde Perl.

La communauté dispose, en plus, du soutien d'entreprises qui participent au développement de PHP, ou qui en sont de fervents supporters. La société Zend propose de nombreuses solutions logicielles autour de PHP, en plus du Zend Engine, interpréteur générique intégré à PHP 4. On peut également citer la société française Nexen, spécialisée dans les services Internet, qui, en plus de ses activités, soutient activement PHP (hébergement de miroirs, diffusion de documentation, etc.).

Quelques liens
Pear (en anglais) :
http://pear.php.net/
CPAN (en anglais) :
http://www.cpan.org
Interview de Rasmus Lerdorf :
http://developpeur.journaldunet.com/itws/it_phpnexen_rasdorf.shtm

1.4. PHP face à ses concurrents (ASP, JSP, etc.)

PHP face aux autres langages de script web

"PHP n'est pas un langage neuf ou révolutionnaire. Il emprunte une grande partie de sa syntaxe à des langages comme le C, Perl ou Java". Qui est l'auteur de ce jugement définitif sur la prétendue révolution ou sur le caractère novateur de PHP ? Bill Gates, Scott McNeally ou Larry Wall ? Aucun d'entre eux ! C'est bien le père de PHP lui-même, Rasmus Lerdorf, dans une interview au journal en ligne *Computerworld*. Et ce n'est pas tout. Quelle est la différence entre PHP et son grand rival propriétaire, ASP, de Microsoft ? Il n'y en a pas vraiment. "Au fond, ils font les mêmes choses", n'hésite pas à ajouter Rasmus Lerdorf. Encore un peu, et l'on va apprendre que M. Lerdorf code en VisualBasic et qu'il en est très content... Non, la curée s'arrête là.

Mais Rasmus Lerdorf a raison. PHP n'est pas une révolution à proprement parler et, sur le fond, PHP n'apporte rien de bien particulier par rapport à ASP, Java ou Perl. Envoyer des mails, générer des pages à la volée, tout cela, PHP le permet tout comme ASP, par exemple.

Comparer PHP à ses grands rivaux n'est pas toujours chose aisée, voire admissible. Ainsi, la comparaison avec Perl n'est pas forcément fondée. Perl est un langage de script, alors que PHP est un langage de script fait pour l'Internet. On ne compare alors qu'une portion de Perl à la totalité de PHP. De même, ASP s'intègre de plus en plus dans la plateforme .Net de Microsoft, ce qui lui ouvre d'autres horizons. PHP, lui, n'est pas inféodé à une plateforme. Idem pour Java. Java a été conçu pour évoluer dans une architecture dite "n-tiers", alors que PHP, lui, reste dans un environnement "2 tiers" (voir schéma).

Se servir de Java pour dynamiser un site personnel, cela revient à quelque chose près à se servir d'un lance-flammes pour désherber un parterre de rosiers !

Ceci étant dit, il est tout de même possible de se pencher sur ces langages de script pour voir quels sont leurs avantages et leurs inconvénients sur les points qu'ils ont en commun.

Comparatif PHP/Perl

Perl est avant tout un langage de script système, né pour remplacer sed et awk. Face à lui : PHP, un langage de script né du Web pour le Web. Les points de comparaison, s'ils existent, ne doivent pas faire oublier que Perl n'a pas pour objectif premier la création de sites web.

De base, PHP inclut beaucoup plus de bibliothèques que Perl. Pour Perl, il faut souvent aller chercher une librairie souhaitée sur CPAN. En revanche, Perl, dispose, au final, de beaucoup plus de bibliothèques que PHP. Là, PHP fait les frais de son jeune âge face à un langage d'âge déjà vénérable. Sans parler des librairies, les solutions en Perl sont déjà très nombreuses. Toutefois, le dynamisme tout particulier de la communauté PHP est en passe de combler cette lacune.

Langage orienté web, PHP est plus facile à maintenir et à faire évoluer que Perl. Retoucher ou adapter rapidement un site tout entier tourné vers Perl reste plus ardu que si le site dispose d'une architecture en PHP.

Perl a aussi su s'adapter au Web. Alors qu'il créait un nouveau processus sur le serveur à chaque nouvelle requête, Perl a, depuis, su normaliser ses rapports avec le serveur web. Le problème a en effet été réglé par l'apparition de mod_perl ou FastCGI.

Comparatif PHP/ASP

ASP (Active Server Pages) est un langage de script développé par Microsoft. PHP et ASP sont assez proches dans leur philosophie et dans leur mise en œuvre. Dans les deux cas, le code du script est inséré directement dans la page web. Jusqu'à PHP 3, ASP disposait d'un très net avantage sur PHP : PHP ne gérait pas les sessions. Avec l'introduction de cette gestion en mode natif dans PHP 4, l'avantage est perdu. Si PHP 3 doit impérativement être utilisé, pour des raisons de licence par exemple, il existe de toute façon des alternatives, comme PHPLib, qui permettent de gérer les sessions malgré tout.

Le gros inconvénient d'ASP par rapport à PHP réside dans sa très forte dépendance à la plateforme Windows. ASP ne se déploie en effet que dans un environnement Microsoft. Il existe certes un portage sous UNIX, grâce à la solution mise en place par Sun avec Chilisoft ASP, mais cela reste en deçà de ce que peut offrir PHP. PHP n'a pas été développé pour une plateforme en particulier, ce qui ouvre de bien plus vastes horizons en matière de portabilité des applications ainsi qu'en évolutivité. PHP fonctionne sous Linux comme sous Windows NT, sans avoir à recourir à des programmes tiers. Un script en PHP pourra être réutilisé bien plus facilement qu'un script en ASP.

Il a souvent été reproché à PHP d'accuser des temps de réponse assez longs. Ce point peut être amélioré grâce à des solutions logicielles de cache. Zend Accelerator ou PHP Accelerator réduisent ainsi nettement les temps de réponse de PHP.

Autre grosse différence entre PHP et ASP : PHP s'interface en natif avec l'ensemble des systèmes de gestion de bases de données standard du marché, alors qu'ASP doit passer par

ODBC. Cette différence peut être envisagée de deux façons. D'un côté, une meilleure maîtrise de la base de données et peut-être même de meilleures performances jouent en faveur de PHP. D'un autre côté, les fonctions varient d'une base de données à l'autre, ce qui nécessite d'adapter le code pour chaque base de données, alors qu'en théorie, ASP ne demandera que quelques modifications. En fait, même si rien n'est proposé en standard, il existe des solutions (présentées dans ce livre) de couches d'abstraction pour PHP, mais, surtout, l'implémentation du langage SQL variant grandement d'un serveur de bases de données à l'autre, l'adaptation du code en fonction du serveur de bases de données est inévitable (c'est d'ailleurs pour cette raison que Rasmus Lerdorf n'est pas favorable à l'intégration d'une couche d'abstraction dans PHP).

Aujourd'hui, ASP va très largement vers .Net, la plateforme "n-tiers" de services web de Microsoft. Il se rapproche en cela de Java avec J2EE de Sun. De son côté, PHP deviendra de plus en plus un langage "sérieux", incluant déclaration de variable, typage, etc.

Face à l'ASP, PHP peut également jouer la carte du prix. Prenons le cas d'une entreprise qui veut développer une solution Microsoft. Windows 2000 Server lui coûtera 934 euros, Internet Security and Acceleration Server, 8 000 euros, SQL Server 2000 Entreprise Edition, 2 300 euros, Microsoft Proxy Server, 900 euros et, enfin, la souscription au MSDN, 2 100 euros par développeur. Ce qui fait un total de 14 234 euros. Face à cela, une architecture similaire autour de PHP peut être construite avec Linux, Apache et SSL, PostgreSQL et Squid, le tout pour zéro euros (hormis le coût de bande-passante requis pour télécharger les produits). Pour une jeune entreprise, une telle manne financière peut se révéler très utile dans d'autres domaines…

Comparatif PHP/JSP

Encore une fois, comparer PHP à JSP est un exercice périlleux. Le principe de fonctionnement est un peu différent : PHP a été conçu pour s'intégrer au code HTML, tandis que JSP est une utilisation de Java permettant de faire gérer des scripts intégrés au code HTML. En fait, une page JSP est systématiquement convertie en servlet. Alors que l'exécution d'un script PHP est composée de l'analyse du document, de la compilation (en opcode) et de l'exécution proprement dite, celle d'un script JSP est composée de l'analyse du document, de la génération de la servlet (qui consiste principalement à convertir le code HTML en appels Java "affichant" ce même code), de la compilation de la servlet et, enfin, de l'exécution proprement dite. Il est possible d'imaginer que c'est la "lourdeur" de ce travail qui fait que les servlets sont systématiquement mises en cache (ce qui crée parfois des surprises aux développeurs débutants), alors que l'utilisation de caches n'est qu'optionnelle avec PHP (il est donc important de tenir compte de cette différence lors d'un comparatif de performances).

Alors qu'avec PHP, tout passe par l'utilisation de scripts PHP, avec JSP, l'utilisation des scripts (scriptlets) doit être réduite au minimum. Ces scriptlets doivent autant que possible se contenter d'appeler des méthodes de Beans. Cela présente l'inconvénient de devoir systématiquement travailler de front dans deux espaces différents : d'un côté les pages web, de l'autres les Beans. Chaque modification entraînera une nouvelle compilation des Beans et, éventuellement, la génération d'une nouvelle archive à déployer dans l'espace du serveur. Avec PHP, une simple modification du script dans l'espace du serveur suffit. En revanche, la compilation des Beans vous assure de ne pas rencontrer de stupides erreurs de syntaxe lors de l'exécution du code (problème que l'on rencontrera avec PHP). Mais ce n'est toutefois pas ça qui vous assurera d'avoir un code qui fonctionne (même s'il est vrai que cela y contribue grandement). En fait, JSP se conçoit comme un élément parmi d'autres dans une architecture J2EE.

Pour des sites de grande envergure, l'avantage est indiscutable : on peut plus facilement partager les tâches et les compétences d'une équipe sur l'élaboration d'un code en Java. De

même, l'architecture "n-tiers" de Java (permettant l'appel d'objets mis à disposition sur un serveur distant) offre un éventail de solutions plus large (et avec des solutions plus robustes) pour des sites à forte audience et trafic dense. Mais pour aboutir à ce résultat, nombreuses sont les normes ou règles de conceptions à respecter ce qui ne facilite pas l'apprentissage du langage. Il est à noter toutefois que bon nombre de ces règles peuvent très bien être appliquées à PHP (comme par exemple le modèle MVC). Dans ce cas, il est juste de la responsabilité de l'équipe de développeurs de s'imposer ou nous ces choix de conception.

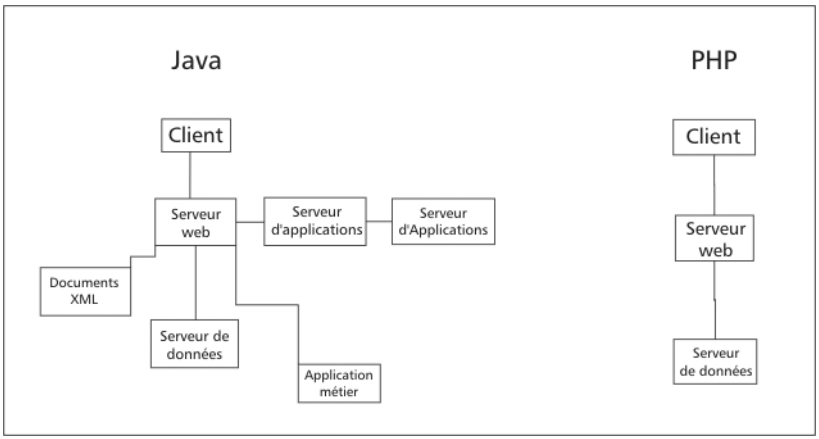


Figure 1.3 : *Architecture en "n-tiers" de Java face à PHP*

Via J2EE, Java rend l'éclatement d'une plateforme applicative beaucoup plus facile et simple que dans le cas de PHP. Pour PHP, le script doit théoriquement être exécuté sur la machine qui reçoit la requête (traitement du script sur la machine qui héberge le serveur web). Aussi est-il plus ardu de mettre en place une structure éclatée, plus modulaire, et tenant de lourdes charges.

L'interfaçage avec les bases de données s'effectue grâce aux pilotes JDBC, ce qui permet de communiquer avec tout type de serveur de bases de données avec un langage commun, mais, comme nous l'avons dit précédemment, ce n'est pas le cas par défaut pour PHP (même si des solutions existent). Toutefois, comme cela a également été dit, le langage SQL varie sensiblement d'un serveur de bases de données à l'autre, rendant nécessaire l'utilisation de code dédié (ce qui réduit grandement l'intérêt d'une telle interface).

Pour une application spécifique, le choix de Java peut se révéler judicieux. Par exemple, on ne peut pas parler de véritable pool de connexion en PHP, alors que, pour Java, il est présent dans le container de EJB-J2EE. Dans le même esprit, il est possible de partager des Beans en commun avec tous les scripts exécutés (ce que ne propose pas PHP).

Il est à noter que Java a été conçu comme étant orienté objet (comme Smalltalk ou ObjectiveC). PHP, lui, n'a pas été créé pour la programmation objet, mais il est tout de même possible de programmer de la sorte, même si toutes les propriétés auxquelles nous pourrions nous attendre ne sont pas disponibles. Aujourd'hui les choses sont en train de changer, et le moteur Zend 2 va plus loin dans le sens de la programmation objet.

Heureusement, comme pour PHP, il existe des solutions open-source plus ou moins complètes comme par exemple Tomcat et JBoss.

Perspectives

PHP a pour lui une très grande facilité d'installation. Il est proposé de base sur quasiment toutes les distributions Linux. De très nombreux kits d'installation existent, qui permettent de faire rapidement ses premiers pas. Ces kits offrent également une solution de mise à jour pratique et facile à mettre en œuvre. PHP fait partie de la fondation Apache, ce qui garantit un interfaçage entre le langage de script et le serveur web, sinon parfait, du moins tenu à jour.

Comme nous l'avons déjà dit à plusieurs reprises, PHP dispose d'une très large et dynamique communauté de développeurs, et les tutoriels en PHP sont très nombreux. Entièrement traduite en français, la documentation officielle est disponible gratuitement sous de nombreux formats. PHP a également su s'inspirer des points positifs de ses concurrents. Le CPAN de Perl avait ainsi donné naissance à son équivalent PEAR (base de ressources centralisées pour PHP).

Il est très largement reconnu que PHP est d'une prise en main facile. Sa syntaxe simple permet d'approcher beaucoup plus vite de la solution.

Pour une entreprise, PHP peut également être intéressant par son faible coût de déploiement. Par rapport à ASP, par exemple, qui sous-entend une plateforme Microsoft avec de nombreuses licences à acheter, PHP aligne des équivalences libres, et bien souvent gratuites, tout aussi fonctionnelles.

Petit point négatif pour PHP, les comparatifs sont bien peu de choses face à une culture d'entreprise. Même si PHP offre, dans certains cas, une solution tout à fait satisfaisante et bien meilleur marché, il sera souvent plus simple de lui préférer une solution ASP ou JSP pour des raisons historiques propres à l'entreprise. C'est là que PHP peut avoir un caractère révolutionnaire. L'adopter, c'est faire confiance à une technologie atypique par son histoire et sa portée. C'est là que le plus gros travail reste à faire pour les développeurs. Si, sur le papier, les décideurs informatiques acceptent bien volontiers les avantages de PHP, basculer une plateforme vers PHP reste une décision importante. La tendance semble pourtant s'inverser du fait du passage de certains grands comptes vers PHP (liberation.fr par exemple) : on fait de plus en plus fait confiance au PHP. Cette confiance associée à une recherche d'économie dans le monde de l'Internet font que c'est tout naturellement que PHP dépasse ASP en terme d'équipement de sites web (avril 2002, sur les sites étudiés par Netcraft).

Documentation officielPHP
http://www.php.net/docs.php

En bref

Pour des langages de script, il est possible de discerner quelques grands facteurs à prendre en compte lors du choix de l'un d'entre eux : la facilité d'apprentissage (accessibilité), la puissance au regard de la complexité, la portabilité et l'environnement disponible (ressources en ligne, outils, etc.). Pour ASP, PHP, Java, Perl et PHP, la situation peut se résumer comme suit :

Tableau 1.1 : Tableau comparatif des langages de script

	PERL	ASP	JSP	PHP
Accessibilité	+	++	-	+++
Puissance	++	++	+++	++
Portabilité	++	-	+++	+++
Environnement	++	++	+++	+++

Au regard des critères que nous avons définis, PHP semble bien avoir le plus d'avantages. Il n'importe pas de savoir si, dans l'absolu, PHP est plus puissant que Java, par exemple. Mais, en revanche, il s'agit de savoir si, dans le cas de l'utilisation d'un langage de script pour rendre un site web dynamique, l'investissement en temps requis pour maîtriser Java est justifiable au regard de la facilité d'usage de PHP. Le *Jargon français* définit les objectifs d'un langage de script comme suit : "Faire simple, rapide, utilitaire". Dans cette optique, PHP est clairement le plus adapté.


INTERNET

Le Jargon
http://www.linux-france.org/prj/jargonf

1.5. Pourquoi ont-ils choisi PHP ?

Ils ont choisi PHP

De nombreuses applications ont déjà été écrites en PHP et les sites webs ("perso" comme professionnels) qui passent au PHP ne se comptent plus. Pourquoi un webmestre fait-il le choix de PHP ? Entre ambition de développement et préférence politique, les atouts de PHP sont variés. De plus, les performances alignées par PHP dans ses multiples applications grand public peuvent également séduire.

Nous avons recueilli les témoignages de deux développeurs qui ont fait le choix de PHP, pour les présenter plus en détail avant de passer à une présentation générale avec l'aide de l'Observatoire français de PHP.

Gros plan : Tuxfamily et DaCode

Tuxfamily, hébergeur indépendant

Lancé par Julien Ducros, Tuxfamily est un hébergeur de projets libres qui utilise PHP au sein de sa plateforme. Agée de trois ans, la structure de Tuxfamily est désormais solide : un système LVS (Linux Virtual Server) orchestré par deux Load Balancer (N.D.A : répartiteur de charge). Toute la plateforme est répliquée et les données utilisateurs sont contenues sur un filer central de 360 Go en raid 5. Les responsables de Tuxfamily connaissent PHP depuis bien longtemps, à l'époque même où l'on parlait encore de PHP/FI.

Pourquoi ont-ils choisi PHP ? Principalement pour sa rapidité de mise en œuvre. Mais ce n'est pas tout : "Le langage PHP a aussi été pour nous un choix politique", explique Pierre Machard. Il poursuit : "PHP est un logiciel libre de grande qualité. Sa flexibilité et sa polyvalence ont motivé ce choix. Par exemple, nous pouvons actuellement l'utiliser indifféremment avec une base de données MySQL ou Postgres". Pour lui, même s'il n'offre pas la puissance des CGI, PHP est beaucoup plus simple à mettre en œuvre et à exploiter. Selon Pierre Machard, la syntaxe de PHP, proche du C, participe aussi de son succès. Enfin, il est, d'après lui, idéal pour épauler Apache.

PHP n'est pas pour autant au-dessus de tout soupçon. Comme toute technologie dite "dynamique", PHP expose les sites à des attaques par *cross site scripting* (exploitation d'une faille de sécurité sur un serveur web en entrant des commandes directement dans l'adresse d'une page). Pour Tuxfamily, il est nécessaire de s'assurer d'avoir la dernière version stable installée. Les alertes de sécurité publiées sont plutôt le signe d'une bonne santé de PHP. En tout cas, "une rustine est rapidement disponible sur l'Internet", nous explique Pierre Machard, "ce qui n'est pas le cas d'ASP, logiciel propriétaire". Pour un site à fort trafic comme Tuxfamily, PHP dispose d'un moteur bien conçu qui "n'accapare pas à outrance les ressources des machines, même s'il possède quelques carences", concède Pierre Machard.

Seul petit bémol adressé à PHP par l'équipe de Tuxfamily : la licence de la dernière version. Les version 3.x étaient libres, alors que la licence appliquée à partir de PHP 4.x est plus restrictive (voir notre partie sur les licences).

daCode, moteur de nouvelles

Créateur et webmestre du site d'informations sur Linux et les logiciels libres **http://linuxfr.org**, Fabien Penso a bien voulu nous expliquer pourquoi il avait choisi PHP pour développer daCode, le moteur de son site.

Il a rencontré PHP par hasard. Cherchant un langage de script simple pour faire des sites web, il a tout naturellement testé PHP. Voulant rendre daCode utilisable par n'importe qui, il a choisi PHP pour développer son logiciel, car c'est le langage disponible sur les plateformes grand public comme Free ou Multimania. Cela allait dans le sens de sa démarche. L'idée étant également d'avoir le plus de contributions possibles pour daCode, l'importante communauté de développeurs PHP a joué en faveur du langage libre. Il fut un temps question de Perl/mod_perl mais cette solution posait trop de problèmes de fuite de mémoire.

Leurs sites
Tuxfamily, hébergement libre pour gens libres :
http://www.tuxfamily.org
Contact : Pierre Machard <pmachard@tuxfamily.org>
Linuxfr, site d'information sur Linux et les logiciels libres :
http://linuxfr.org
daCode, gestionnaire de contenus open source (GPL) :
http://www.dacode.org
Contact : Fabien Penso <penso@linuxfr.org>

PHP à l'assaut du Net

On ne dénombre plus les grands comptes qui se tournent vers PHP. Que ce soit par la petite porte, soufflé à un directeur informatique par un développeur, ou dans un plus large panel d'offres faites par une webagency, PHP n'est plus un langage neuf ou exotique. Les décideurs en informatique lui font de plus en plus confiance.

Le second trimestre 2002 a vu naître deux structures parallèles visant à recenser toutes les utilisations professionnelles ou industrielles de PHP. L'Association française des utilisateurs de PHP (AFUP), née en avril 2002, et l'Observatoire français de PHP (OFPHP), né en juin 2002, présentent des interviews, des études et des listes de sites utilisant PHP. Le surf sur ces sites montre à quel point PHP a pénétré les hautes sphères du web.

Sur le site de l'AFUP, on apprend que le quart des entreprises du CAC 40 utilisent PHP pour leur site web. Parmi elles, Cap Gemini, PSA Peugeot Citroën, Schneider Electric SA, etc. On le voit, pour les grands comptes, le langage de script libre fait très largement jeu égal avec ASP, JAVA ou ColdFusion.

Du côté de l'OFPHP, la liste des sites utilisant PHP de par le monde, présentée dans la section *Références*, est impressionnante : http://sourceforge.net (30 millions de pages visionnées par mois), http://www.phpbuilder.com (9 millions de pages par mois) ou encore http://www.insight.com (plus de 2 milliards $US. de chiffre d'affaire en 2000). Pour la France, citons simplement http://www.boursorama.com (14 378 807 visites en juillet 2001 avec une pointe à 9 783 039 pages vues pour la seule journée du 17 septembre 2001, jour de la réouverture des marchés à Wall Street). L'OFPHP présente également des interviews d'acteurs de l'internet qui ont choisi de faire confiance au PHP.

L'utilisation de PHP dans les entreprises
AFUP : le quart des entreprises du CAC 40 utilisent PHP
http://www.afup.org/article.php3?id_article=105
OFPHP :
http://www.ofphp.com/index.php?page=references
ZDNet : les logiciels libres tiennent la charge
http://techupdate.zdnet.fr/story/0,,t381-s2111391,00.html

Chapitre 2

Prise en main

Pour créer un site en PHP, il n'y a pas besoin de beaucoup de choses. Un éditeur de texte et un serveur web suffisent.

Si vous utilisez les services d'un hébergeur Internet, vous pouvez éventuellement vous contenter de déposer vos scripts dans l'espace qui vous est réservé et voir le résultat obtenu. Mais il est bien plus pratique d'installer son propre serveur web sur sa machine, afin de tester ses scripts avant d'aller les déposer chez un hébergeur.

Autrement dit, quels que soient les cas, vous serez amené à installer un serveur web supportant PHP. D'ailleurs, vu la simplicité de l'opération, pourquoi s'en priver ?

Si vous êtes votre propre hébergeur, le chapitre *Configuration* vous permettra de personnaliser les options autorisées. Dans le cas contraire, il est conseillé de consulter ce chapitre qui vous permettra (via un appel à `phpinfo()`) de mieux connaître ce que permet votre hébergeur et ce qu'il interdit.

Enfin, ce chapitre vous présente quelques éditeurs particulièrement adaptés à PHP.

2.1. Installation

PHP est disponible pour différents serveurs et systèmes d'exploitation. Dans ce chapitre, nous nous efforcerons de décrire au mieux l'installation de PHP sur les serveurs Apache, IIS, iPlanet, etc.

Avec Apache

PHP pour Apache est notamment disponible sous les environnements Linux/UNIX, Windows et Mac OS X.

Apache 2
Apache 2 est considéré par ses développeurs comme la version officielle du serveur web. Malheureusement, Apache2 n'est pas encore officiellement supporté par les développeurs de PHP. Dans certains cas, la version 5 de PHP peut fonctionner avec Apache 2. Dans tous les cas, consultez la documentation officielle pour connaître l'évolution de la compatibilité : **www.php.net/manual/fr/install.apache2.php**. *Nous présenterons donc essentiellement l'installation avec la première version d'apache tout en indiquant les différences mineures pour une installation avec Apache2.*

Sous Linux/Unix

Nous pouvons considérer qu'il existe trois grandes méthodes d'installation de PHP dans un environnement Linux/UNIX. Cela peut ainsi se faire :

- En utilisant les scripts d'installation fournis avec les distributions (Linux notamment) ;
- En utilisant des kits d'installation disponibles sur Internet ;
- En faisant une installation "manuelle".

Utiliser les scripts d'installation de votre distribution est le moyen le plus sûr et le plus rapide pour débuter avec PHP, mais passer par une installation "manuelle" vous permet de personnaliser au mieux votre serveur.

Installation via la distribution Linux (ou UNIX)

Lors de l'installation du système d'exploitation, de nombreuses distributions offrent à leurs utilisateurs la possibilité d'installer un serveur Apache avec PHP et, bien souvent, MySQL, voire également d'autres bibliothèques. Il suffit alors généralement de cocher une case ou deux. Certaines distributions proposent même des versions dites "optimisées".

Nous ne pouvons malheureusement pas toutes les détailler ici. Vous êtes donc convié à consulter la documentation fournie avec votre système d'exploitation.

Il est également possible d'installer cet ensemble Apache/PHP (et éventuellement MySQL) a posteriori, grâce aux systèmes des "packages" (parfois appelés paquetages) portant généralement l'extension *.rpm*.

Installation via un kit

Si votre distribution ne propose pas de quoi installer simplement PHP dans votre environnement, ou bien si elle ne propose pas de version récente des serveurs Apache et de PHP, vous pouvez utiliser un kit d'installation.

Il existe, par exemple, Apache Toolbox, que vous trouverez à l'adresse **http://www.apachetoolbox.com** (mais malheureusement en anglais). ApacheToolbox est un projet qui permet en effet d'installer et de mettre à jour très facilement Apache, PHP (version 3 ou 4, au choix), MySQL, ZendOptimizer, OpenLDAP, ainsi que près de cinquante modules pour Apache.

Bien que les auteurs ne l'aient jamais essayé, sachez qu'il existe LinuxEasyInstaller, disponible à l'adresse **http://www.phpmylinux.net/index.php3?rub=lei**, qui installe Apache/PHP, MySQL et phpMyAdmin (scripts permettant l'administration des bases de données via un navigateur).

Installation manuelle

L'installation manuelle est certainement la meilleure façon d'avoir exactement ce que l'on veut : Apache et PHP avec la base de données de son choix (PostgreSQL ou Oracle, par exemple) et pas nécessairement MySQL, ainsi qu'avec les bibliothèques que l'on souhaite (au choix parmi celles présentées tout au long de ce livre) et pas seulement avec celles supportées par les kits d'installation (que ce soient ou non ceux des distributions). C'est également le plus sûr moyen d'avoir la toute dernière version (chacun des éléments étant librement téléchargeable sur Internet).

Bref, même si ce n'est pas la façon la plus simple d'installer PHP, elle reste celle que nous préconisons. Les autres méthodes présentées précédemment sont toutefois fort pratiques pour débuter en PHP.

Pré-requis

Il va de soi que, pour installer Apache manuellement, vous devez avoir une connaissance des bases de Linux (la notion de droit utilisateur) et des commandes élémentaires que sont `cp`, `cd` ou `ls` par exemple. Nous ne reviendrons pas sur l'utilisation du système Linux, et encore moins sur son installation. En revanche, vous trouverez d'excellentes documentations sur Internet ou

dans des livres. Ce message d'avertissement n'a pas pour objectif de vous décourager ou de vous effrayer devant cet excellent système d'exploitation qu'est Linux. Nous vous invitons simplement à prendre connaissance des bases avant de vous lancer plus avant dans l'installation d'Apache et de PHP.

Installation

La notice d'installation suivante fonctionne avec la plupart des distributions Linux (RedHat, Mandrake, SuSE, etc.). Il peut néanmoins y avoir des différences d'arborescence, les différentes distributions n'ayant pas toujours la même logique. Pour cela, reportez-vous au manuel de référence de votre système d'exploitation, qui doit vous fournir une description détaillée de sa structure. Il vous suffit alors de modifier les chemins donnés dans les lignes de commande. Les grandes procédures, elles, restent inchangées.

L'installation se réalise en deux temps :

1. Dans un premier temps, vous devez installer les bibliothèques que vous souhaitez utiliser.
2. Vous devez compiler PHP en tenant compte des bibliothèques précédemment installées, puis compiler Apache en intégrant PHP.

Suite de l'installation
Une fois connecté et authentifié sur un système Linux, et pour que la suite de l'installation se déroule correctement, vous devez passer en mode super utilisateur (`root`). Pour cela, dans un shell, tapez simplement la commande "`su -`" suivie du mot de passe `root` quand le système vous le demande.

Installation des bibliothèques

Si vous ne savez pas encore quelles bibliothèques vous allez utiliser, vous pouvez vous contenter d'une installation de base (sans bibliothèque particulière).

Les procédures d'installation des bibliothèques sont décrites dans le chapitre traitant de la bibliothèque, mais, généralement, le principe est le suivant :

il suffit de télécharger la dernière version de la bibliothèque sur Internet (ou d'utiliser celle disponible sur le CD-ROM), de copier l'archive (par exemple sous */usr/local/src*), et de taper les commandes suivantes :

```
# tar zxvf librairie-version.tar.gz
# cd librairie-version
# ./configure
# make
# make install
```

Si vous rencontrez un problème, pensez à consulter le fichier *INSTALL* (bien souvent en anglais) qui doit être fourni avec l'archive d'installation. Vous y trouverez la documentation complète pour l'installation de la librairie, qui nécessite peut-être des paramètres spéciaux. Soyez également attentif au message d'erreur qui pourrait être donné par le compilateur. Il constitue généralement un excellent moyen de savoir où se situe le problème (dépendances, version de compilateur trop ancienne, etc). Le fichier *README* (également en anglais, à moins

que vous trouviez un *LISEZMOI*) peut également contenir des informations utiles comme, par exemple, des incompatibilités temporaires.

Compilation de PHP et Apache

Vous devez, dans un premier temps, télécharger les dernières versions d'Apache et de PHP (ou utiliser celles fournies sur le CD-ROM).

Les sites officiels
Apache :
http://www.apache.org
PHP :
http://www.php.net

Autant que possible, téléchargez les sources au format *.tar.gz* qui seront utilisées dans la suite de notre explication. Cela vous permettra d'avoir des binaires optimisés pour votre système.

Allez dans le dossier contenant les archives que vous avez téléchargées. Vous pouvez d'ores et déjà passer en mode administrateur (`root`), puisque cela est généralement nécessaire pour l'opération d'installation. Décompressez alors les deux archives (nous supposerons ici que les archives sont décompressées dans le même répertoire).

```
# tar zxvf apache_1.3.31.tar.gz
```

Dans le cas d'apache 2, la commande devient `tar zxvf httpd-2.0.50.tar.gz`.

```
# tar zxvf php-5.0.1.tar.gz
```

Pour compiler PHP en tant que module APXS, vous devrez d'abord compiler Apache en vous assurant qu'il autorise le chargement des modules dynamiques.

```
# cd httpd_X
# ./configure --prefix=/usr/local/apache --enable-module=so
# make
# make install
```

Apache est désormais installé sous */usr/local/apache*.

Reste à compiler le module APXS PHP.

```
# cd ..
# cd php-5.0.1
# rm config.cache
```

La suppression du fichier *config.cache* est inutile lors de la première compilation (ce fichier n'existant pas), mais, par la suite, elle est quasiment indispensable si vous souhaitez être sûr de recompiler PHP avec les nouvelles options précisées.

Vous pouvez alors passer à la configuration (préparation à la compilation de PHP).

```
# ./configure --with-apxs=/usr/local/apache/bin/apxs <autres options>
```

Dans le cas d'apache 2, la commande devient `./configure --with-apxs2=/usr/local/apache/bin/apxs <autres options>`

Il existe de nombreuses options de configuration consultables en tapant `./configure --help`. Pour chaque bibliothèque présentée dans ce livre, l'option de configuration est indiquée (vous devrez sans doute en cumuler plusieurs). Ainsi, par exemple, la compilation de PHP avec `mysql` et `gd` donnera :

```
# ./configure --with-apxs=/usr/local/apache/bin/apxs --with-mysql=/usr/local/
✂ mysql --with-gd=/usr/local --with-jpeg-dir=/usr/local --with-png-dir=/usr/
✂ local --with-zlib-dir=/usr/local
```

Il est à noter qu'il n'est pas obligatoire de préciser les chemins des bibliothèques, PHP s'assurant de retrouver la bibliothèque concernée Mais il est toutefois conseillé de le faire afin de s'assurer qu'il n'y ait pas de malentendu (en particulier sur la version à installer).

Vous pouvez ensuite passer à la compilation :

```
# make
```

Compilation sous Solaris
Sous Solaris, vous devrez vous assurer de compiler PHP avec les outils GNU. En d'autres termes, vous devrez installer le "package" binutils (disponible sur le site ***http://www.sunfreeware.com****) et vérifier que le make GNU est bien le premier trouvé dans le* `PATH`*.*

```
# make install
```

Ça y est, PHP est compilé et installé (un fichier *libphp5.so* a été ajouté dans le répertoire *libexec* d'Apache). Reste à configurer PHP et Apache. Outre ce qui est décrit dans le paragraphe suivant, vous devrez consulter le fichier */usr/local/apache/conf/httpd.conf* et vérifier qu'il contient bien la ligne suivante (de préférence juste après les lignes correspondant à la même directive). Au besoin, ajoutez la :

```
LoadModule php5_module libexec/libphp5.so
```

Compilation de PHP intégré à Apache
Il est également possible de compiler PHP au sein d'Apache (sans utiliser de module). Pour cela, la procédure devient :

```
# cd apache_1.3.31
# ./configure
# cd ..
# cd php-5.0.1
# rm config.cache
# ./configure --with-apache=../apache_1.3.31 <autres options>
# make
# make install
# cd ../apache_1.3.31
```


REMARQUE

```
# ./configure --prefix=/usr/local/apache
✂ --activate-module=src/modules/php5/libphp5.a
# make
# make install
```

Dans ce cas, vous ne devrez pas ajouter ou devrez commenter la ligne :

```
LoadModule php5_module libexec/libphp5.so
```

Reste à copier le fichier de configuration par défaut de PHP dans le répertoire où il doit se trouver :

```
# cd ../php-5.0.1
# cp php.ini-dist /usr/local/lib/php.ini
```


RENVOI

Les principales options de php.ini seront décrites dans le prochain chapitre.

Ensuite, éditez le fichier */usr/local/apache/conf/httpd.conf* à l'aide de Vi ou Emacs (selon votre appartenance), et ajoutez la ligne suivante afin que les fichiers portant les extensions indiquées soient interprétés par PHP :

```
AddType application/x-httpd-php .php .php5 .php4 .phtml
AddType application/x-httpd-php-source .phps
```


REMARQUE

Pour voir la vie en rose (ou bleu, ou jaune, ou rouge...)
Si vous donnez une extension .phps à vos scripts, alors ceux-ci ne seront pas interprétés, mais le code source sera affiché et colorisé, ce qui peut parfois permettre de voir comment PHP l'analyse (et ainsi détecter des parse error).

Il est aussi préférable de modifier l'instruction suivante :

```
DirectoryIndex index.html
```

en :

```
DirectoryIndex index.html index.php index.php5 index.php4
```

Cette instruction permet de forcer la page index que le navigateur doit afficher si l'URL demandée est incomplète. Avec la ligne précédente, le serveur ira chercher dans le répertoire précisé le fichier *index.html* et, s'il ne le trouve pas, *index.php*, etc. Ainsi, appeler la page **http://localhost/** renvoie la page se trouvant à l'URL **http://localhost/index.html**, si celle-ci existe.

Pensez également, si ce n'est déjà fait, à décommenter ou ajouter une ligne

```
ServerName <nom de votre machine>
```

Vous pouvez par exemple indiquer :

```
SeverName localhost
```

À présent, lancez le serveur Apache :

```
/usr/local/apache/bin/httpd start
```

Pour l'arrêter, vous n'aurez qu'à taper la commande :

```
/usr/local/apache/bin/httpd stop
```

Si vous n

Vous pouvez maintenant créer votre première page web. Éditez la page */usr/local/apache/htdocs/index.php* et insérez-y ces quelques lignes :

```
<?php
phpinfo();
?>
```

Vous n'avez plus qu'à appeler la page depuis votre navigateur en entrant l'adresse **http://localhost/index.php**.

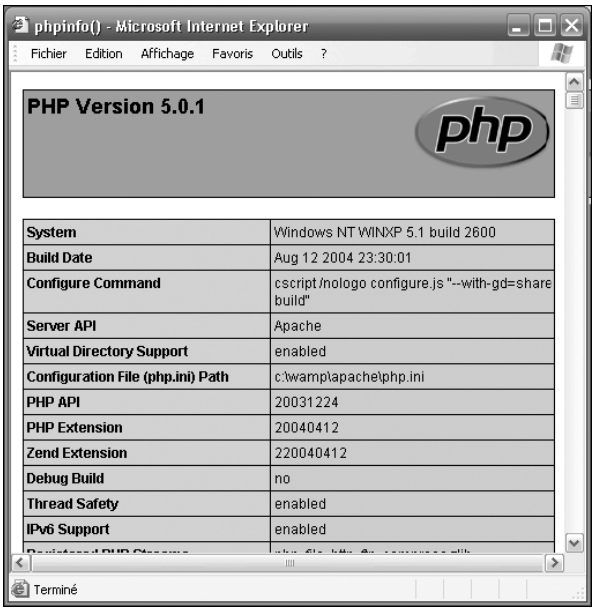


Figure 2.1 : *Voilà votre page interprétée en PHP*

Sous Windows

Tout comme pour Linux, l'installation sous Windows pourra se faire de deux façons différentes : en cliquant ou en compilant. Il existe désormais de nombreuses solutions "tout en un" qui vous permettent d'installer Apache et PHP sur un système Windows sans avoir à faire de fastidieuses manipulations. De plus, ce type d'installation ne nécessite qu'un minimum de temps et de connaissances techniques. Si ces programmes ont pour eux la simplicité et la rapidité, ils ont aussi les défauts évidents de leurs avantages : installation non transparente et pas toujours personnalisable à souhait. Il faut parfois également aller ajouter "manuellement" des bibliothèques.

Installer PHP5 pour Apache 2
Si vous souhaitez installer PHP5 et Apache2, vous devrez télécharger l'archive zippée de PHP5. L'installeur automatique ne vous fournira pas tous les fichiers requis pour que PHP5 fonctionne avec Apache2.

Installation automatique

EasyPHP

EasyPHP est un programme bien pratique, qui permet d'installer Apache, PHP, MySQL et PhpMyAdmin en quelques clics de souris. Il y a environ une nouvelle mouture chaque année.

Pour débuter l'installation, lancez l'exécutable *EasyPHP1-7_setup.exe* présent sur le CD-ROM. Cela installera Apache 1.3.27, PHP 4.3.3, MySQL 4.0.15 et Phpmyadmin 2.5.3. Vous pouvez également vous rendre sur le site d'EasyPHP pour vérifier si une version plus récente est disponible (www.easyphp.org). Si c'est le cas, il est vivement conseillé d'utiliser le produit le plus récent.

Figure 2.2 : *Début de l'installation d'EasyPHP*

Quoi qu'il en soit, l'installation n'est en rien hors du commun, et se déroule sans difficulté particulière. Une fois l'opération menée à terme, le programme d'installation vous propose de vous rendre sur la page d'accueil d'EasyPHP.

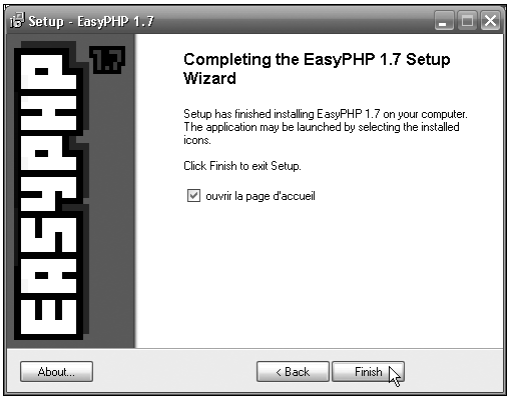


Figure 2.3 : *Fin de l'installation d'EasyPHP*

Ne changez rien, puis cliquez sur **Terminer** (enfin... **Finish** si comme dans votre environnement vous avez un texte mi-anglais mi-français) Et voilà, c'est fini. Un navigateur s'ouvre alors pour vous présenter la page d'accueil d'EasyPHP.

Si le logiciel ne se lance pas automatiquement, dans le menu **Démarrer**, cliquez sur *Tous les programmes*, puis sur *EasyPHP 1.7* et, enfin, sur *EasyPHP*. Le programme va alors démarrer.

Figure 2.4 : *Page d'accueil d'EasyPHP*

La page d'accueil d'EasyPHP présente plusieurs choses fort utiles : une introduction à EasyPHP en local et des liens vers le support en ligne. Vous pouvez également accéder à cette page en cliquant droit sur le "E" qui a été ajouté à la barre des tâches (déjà certainement bien assez chargée à votre goût).

Figure 2.5 : *Élément de la barre des tâches*

Ce *E* vous permettra d'accéder aux principales fonctions relatives à l'administration d'un serveur web.

Dans le menu contextuel qui s'ouvre alors, il suffit de choisir *Web local*, mais, pour le moment, cliquez plutôt sur *Administration*. Cette option vous permet d'accéder à une configuration centralisée de certains des programmes que vous venez d'installer.

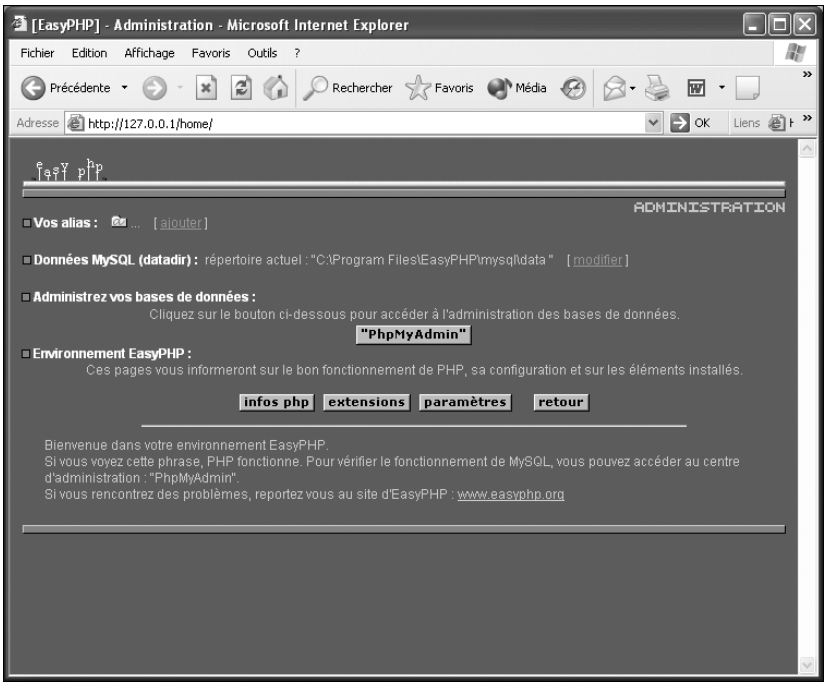


Figure 2.6 : *La page d'administration d'EasyPHP*

Il vous est alors possible (par exemple) de vérifier puis de changer le mot de passe d'accès à MySQL depuis le compte "`root`".

Dans la section *Environnement EasyPHP*, cliquez sur le bouton **Paramètres**. La page se rafraîchit pour vous donner les informations relatives aux paramètres de connexion par défaut attribués à votre serveur MySQL. Vous devez lire quelque chose comme ceci :

```
Paramètres par défaut de la base de données :
serveur : "localhost"
username : "root"
mot de passe : ""
```

Pour le serveur "`localhost`", l'utilisateur "`root`" (celui qui a tous les droits) n'a pas de mot de passe ; il peut se connecter sans même avoir à s'identifier. Il peut donc être important de corriger cela (si plusieurs personnes ont accès à votre machine). Toujours sur la même page, cliquez sur **PhpMyAdmin** dans la section *Administrez vos bases de données*. Vous devez voir s'afficher la page d'accueil de PhpMyAdmin dans un nouveau navigateur. Pour plus d'informations, vous pouvez vous reporter à la partie des annexes traitant de PhpMyAdmin.

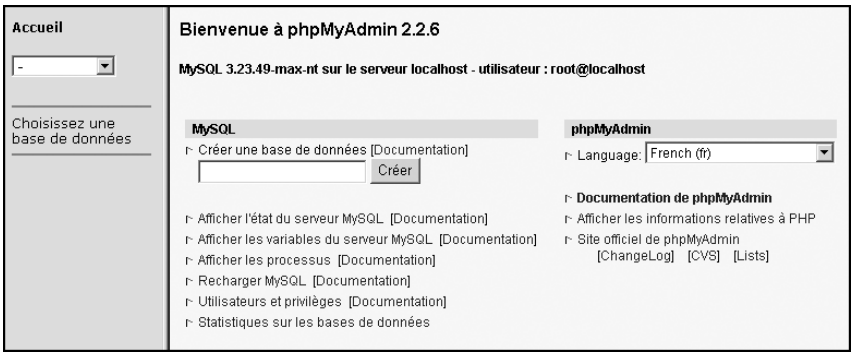


Figure 2.7 : *PhpMyAdmin*

Dans la partie gauche de la page, vous devez trouver, sous la mention *Accueil*, un menu déroulant des différentes bases de données présentes sur le système. La partie qui nous intéresse se trouve sur la partie droite de la page. On peut voir deux colonnes : une première qui propose des liens en rapport avec MySQL, et une seconde qui concerne PhpMyAdmin. Dans la colonne *MySQL*, cliquez sur le lien *Utilisateurs et privilèges*. La nouvelle page qui s'affiche présente alors un tableau récapitulatif des utilisateurs ayant un accès au serveur, un résumé de leurs privilèges, ainsi que diverses options concernant les utilisateurs. Sur la ligne de l'utilisateur "`root`", cliquez sur le lien *Modifier*. La page se recharge. Certains des champs ont été pré-remplis. Vérifiez que le champ "*Utilisateur*" contienne bien "`root`", puis donnez par deux fois le mot de passe que vous souhaitez voir utilisé pour cet utilisateur. Cliquez enfin sur *Exécuter*. Voilà. N'oubliez pas de noter le mot de passe de l'utilisateur "`root`".

N'oubliez pas de lancer EasyPHP lorsque vous voulez utiliser Apache ou MySQL.

Autre intérêt d'EasyPHP : il peut servir de base de départ. Rien ne vous empêche, sans avoir à tout réinstaller, de mettre à jour certains des programmes mis à disposition par EasyPHP. Cela se fera, certes, au prix de quelques manipulations. PHP lui-même pourra, par exemple, être mis à jour facilement grâce à PHPInstaller.

WAMP5 : pour installer PHP5, MySQL4 et Apache2
WAMP5 est un système similaire à EasyPHP qui permet d'installer PHP5 et MySQL 4.0.18 automatiquement. Apache2, lui, est disponible sous forme d'add-on. Vous pouvez trouver WAMP5 à cette adresse : **www.wampserver.com**. *Attention toutefois : si vous avez déjà installé Apache2 ou PHP5, il faudra les désinstaller sans quoi il risque d'y avoir un conflit avec les versions installées par WAMP5.*

PHPInstaller : mettre PHP à jour facilement

PHPInstaller est un programme du groupe PHP (**http://www.php.net**), qui vous permet d'installer PHP sur votre système. Il est pratique à plus d'un titre. Si vous disposez déjà d'un serveur web sur votre machine, il lui est possible de configurer lui-même ce serveur. Malheureusement, le programme de configuration automatique n'existe pas encore pour tous les serveurs web, et il faudra donc mettre la main à la pâte dans certains cas (Apache, tout particulièrement). PHPInstaller est donc plutôt a réserver pour une première installation avec un serveur IIS ou dans

les autres cas, pour une mise à jour. Ceci constitue un moyen simple et rapide de suivre au plus près l'évolution de PHP.

Pour commencer, lancez l'exécutable *php-5.0.1-installer.exe* présent sur le CD-ROM.

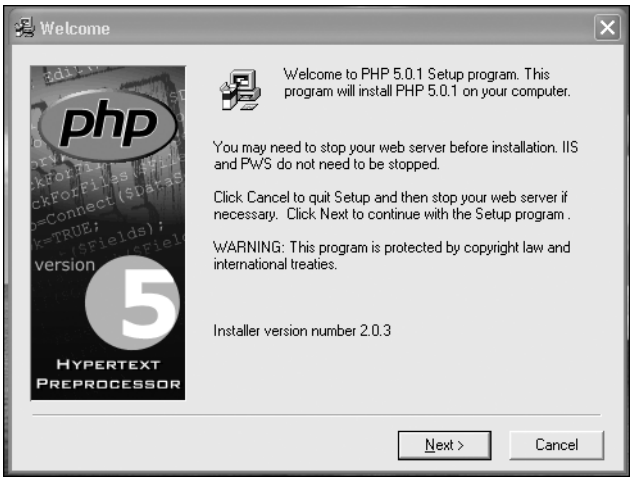


Figure 2.8 : *Page d'accueil de PHPInstaller*

Une fois passée la page d'accueil et celle concernant la licence, PHPInstaller va vous proposer deux types d'installation.

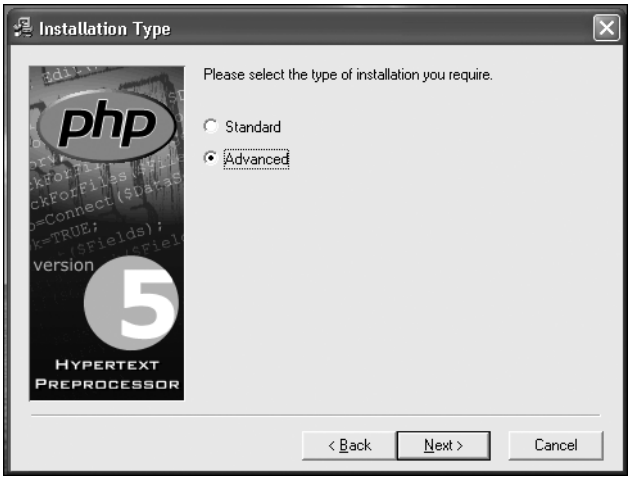


Figure 2.9 : *Choix du type d'installation*

Nous allons décrire l'installation avancée, l'installation standard n'étant pas assez complète pour garantir un minimum de personnalisation. Une fois le processus démarré, répondez aux questions que vous pose l'Assistant. Choisissez un répertoire d'installation, puis acceptez la sauvegarde des fichiers déjà présents sur le système, comme vous le propose PHPInstaller.

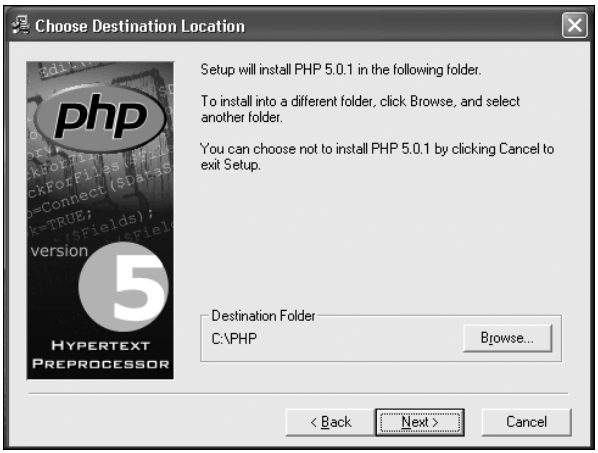


Figure 2.10 : *Répertoire d'installation*

Les options qui arrivent maintenant sont propres au PHP. Vous devez donner un répertoire temporaire qui servira pour les fichiers en attente de publication ("*files upload*") ainsi qu'un répertoire de stockage pour les fichiers de sessions. Ensuite, lorsque la fonction `mail()` sera utilisée, vous devrez spécifier l'adresse qui sera utilisée pour le champ "*de*" ("*from*").

Figure 2.11 : *Configuration du mail*

Vous pouvez ensuite choisir le niveau de rapport d'erreur. Votre choix sera guidé par l'utilisation que vous comptez faire du serveur sur lequel vous êtes en train d'installer PHP. S'il a pour vocation de devenir un serveur de test, vous choisirez l'affichage maximal, cela afin de cerner au mieux les erreurs et dysfonctionnements. En revanche, si vous installez PHP sur une machine qui hébergera un site professionnel destiné au grand public, choisissez le plus bas niveau d'affichage. Les visiteurs ne sont pas concernés par ces questions de programmation, et il n'est pas nécessaire de faciliter la tâche à d'éventuels pirates qui sauraient alors directement où aller fouiller...

Ensuite, choisissez le serveur HTTP sur lequel vous êtes en train d'installer PHP : Microsoft PWS, IIS 3 ou 4, Apache ou Xitami.

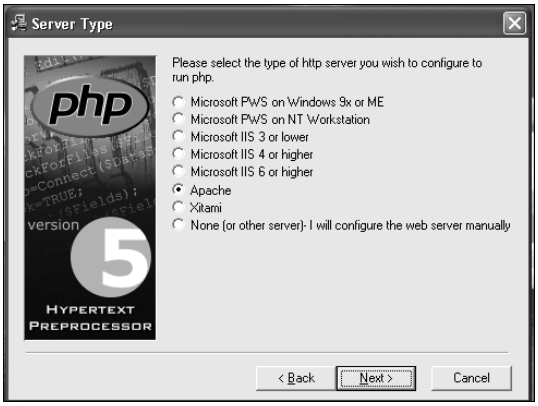


Figure 2.12 :
Sélection du serveur Web

Si le serveur que vous utilisez ne se situe pas dans cette liste, vous devrez alors passer par la case "configuration à la main post-installation" (vous pouvez d'ailleurs utiliser les différentes méthodes présentées dans ce livre).

Enfin, choisissez les extensions qui devront être interprétées comme étant de PHP. Cette dernière étape passée, PHPInstaller passe à l'installation des composants de PHP sur votre machine. Si vous disposiez déjà de PHP sur votre machine, le programme vous demandera si vous souhaitez ou non conserver votre ancien *php.ini*. Cela peut s'avérer très intéressant si vous aviez personnalisé votre fichier *php.ini*.

Figure 2.13 :
Voilà, c'est fini !

Et voilà, c'est fait.

REMARQUE

Mise à jour de PHP pour une installation d'EasyPHP
Dans le cas d'une mise à niveau de PHP pour un système sur lequel avait été installé EasyPHP, il faudra jongler un peu avec le fichier php.ini. EasyPHP installe PHP dans un répertoire qui lui est propre. Si vous désirez conserver la configuration d'EasyPHP, ce qui est recommandé, vous devrez éditer le fichier php.ini situé à la racine de votre dossier Windows pour lui donner les bons chemins pointant vers les différents composants de PHP (php.exe, dossier des extensions, etc.).

Installation personnalisée

Vous pouvez choisir d'installer les différents éléments de la configuration un par un, grâce à des solutions plus ou moins automatisées. Voici comment procéder.

Installer Apache

Il existe des programmes qui vous permettent d'installer facilement la dernière version d'Apache sur votre machine. Téléchargez l'un de ces programmes, de préférence depuis l'un des miroirs du site Apache (http://mir2.ovh.net/ftp.apache.org/dist/httpd/binaries/win32/). Vous pouvez choisir entre un installateur automatique en *.exe* ou en *.msi* (Microsoft Installer). Si vous optez pour la version en *.msi*, assurez-vous que vous disposez bien de l'installateur Windows. Pour vérifier, dans le menu **Démarrer**, puis dans *Exécuter*, tapez la commande **msiexec**. Il vous faut au minimum la version 1.10.1029.1. Si vous ne disposez pas de la version appropriée, téléchargez le programme depuis l'une de ces adresses, en fonction de votre plateforme :

- Windows NT 4.0 : www.microsoft.com/downloads/release.asp?ReleaseID=17344
- Windows 95 et 98 : www.microsoft.com/downloads/release.asp?ReleaseID=17343

Une fois que vous avez téléchargé le programme d'installation d'Apache, lancez-le.

Figure 2.14 : *Fenêtre d'accueil*

Acceptez la licence du programme, puis cliquez sur OK jusqu'à ce que vous arriviez sur la page qui vous pose quelques questions relatives à la configuration du serveur.

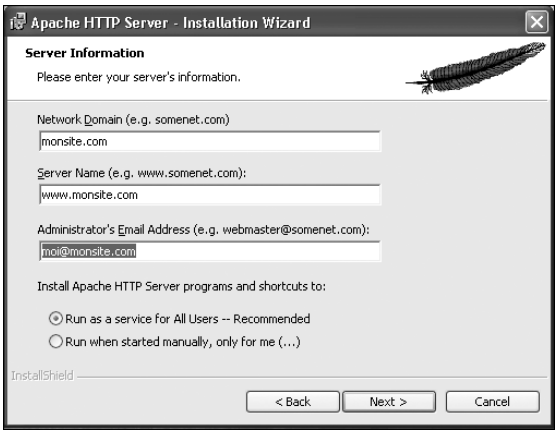


Figure 2.15 : *Paramétrage minimal*

Remplissez les champs *Network Domain* (nom du domaine pour lequel le serveur est installé), *Server Name* (nom du serveur web qui va être installé), et donnez l'adresse e-mail de l'administrateur. Choisissez ensuite si Apache doit être installé comme un service utilisable pour tous les utilisateurs, ou s'il doit être lancé manuellement par un utilisateur donné.

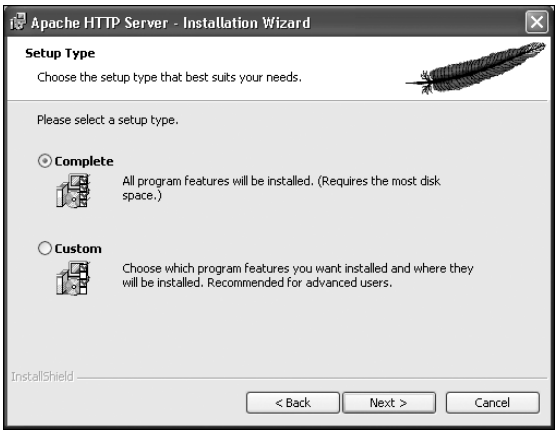


Figure 2.16 : *Choix du type d'installation*

À vous de faire votre choix selon les besoins du serveur que vous mettez en place. Donnez ensuite le chemin d'installation pour que le processus puisse arriver à terme.

Figure 2.17 : *Fin de l'installation*

Par la suite, vous pourrez réutiliser cet installateur pour réparer votre serveur web Apache si vous l'avez mis à mal avec de trop nombreux tests. Il vous suffit de relancer l'installateur pour qu'il vous propose soit de supprimer Apache, soit de le réparer.

Ruse de sioux

Si vous installez Apache pour faire des tests sur votre machine personnelle, voici comment renseigner les champs de l'installateur Apache à l'étape Server Information.

- *Network Domain : localdomain ;*
- *Server Name : localhost.localdomain ;*

- *Administrator's Email Address : votre adresse e-mail.*

Si vous utilisez Windows XP avec le Service Pack 2, vous aurez certainement droit à une alerte de sécurité à la fin de l'installation d'Apache.

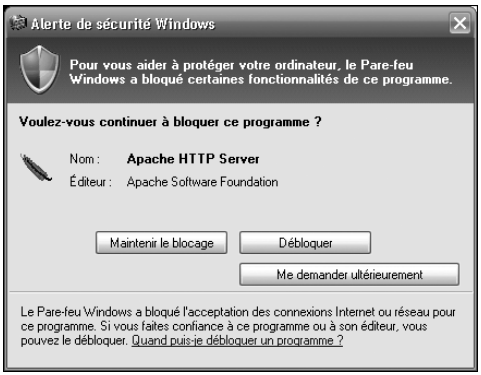


Figure 2.18 : *Alerte de Windows XP lors du démarrage d'Apache*

Cliquez sur le bouton **Débloquer** pour permettre l'exécution du serveur web.

Une fois Apache installé, il va se lancer automatiquement. Vous pourrez trouver une petite plume ornée d'une flèche vers la droite dans la barre des tâches Windows qui permet d'accéder à la console de gestion de votre serveur Web.

Figure 2.19 : *Cliquez sur la petite flèche verte sur fond blanc pour accéder à la console de gestion Apache*

La console de gestion vous permet de connaître le statut du serveur, de l'arrêter, de le relancer ou bien d'accéder à la console de gestion des services Windows.

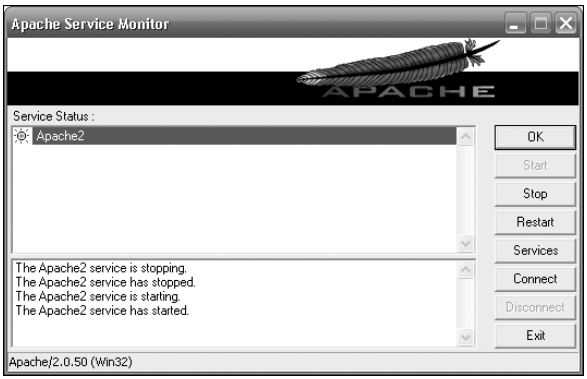


Figure 2.20 : *La console de gestion du serveur Apache (ici c'était un test avec Apache2)*

Vous pouvez lancer un navigateur et le faire pointer sur l'adresse **http://localhost** pour vérifier si la page de test Apache s'affiche bien.

Une fois toutes ces étapes effectuées, vous pouvez passer à l'installation de PHP.

Installer PHP

Téléchargez l'archive zippée disponible sur le site du groupe PHP (*PHP Zip package* dans la partie *Windows Binairies* de la section *Download* du site). Une fois que vous avez téléchargé le fichier compressé, décompactez-le sur votre disque dur dans un dossier spécialement créé à cet effet. Sans vouloir vous influencer, *c:\php* serait une bonne idée (évitez tout particulièrement les noms de dossiers avec des espaces). Dans ce dossier, repérez le fichier *php.ini-dist*, renommez-le en *php.ini*, et copiez-le dans votre répertoire Windows (*c:\Windows* ou *c:\WINNT*). Une fois le fichier copié, éditez-le avec votre logiciel préféré.

Recherchez cette ligne :

```
; extension_dir =
```

et renseignez le champ avec l'emplacement où se trouvent les extensions PHP. Par exemple :

```
extension_dir = c:\PHP\ext\
```

si vous avez installé PHP dans un dossier PHP à la racine de votre disque dur. N'oubliez pas, pour cette ligne et pour toutes les autres, de les décommenter lorsque vous aurez rentré les paramètres nécessaires. Pour cela, il vous suffit de supprimer le "`;`" en début de ligne. Attention, ne décommentez que la ligne contenant des paramètres ; laissez les lignes d'explication en commentaire.

Recherchez ensuite la ligne

```
doc_root =
```

et renseignez le champ avec l'emplacement de la racine de votre serveur web. Il s'agit en fait de donner le chemin d'accès du dossier *c:\Program Files\Apache Group\Apache\htdocs*.

```
doc_root = "C:\Program Files\Apache Group\Apache\htdocs"
```

Une fois que vous en avez terminé avec *php.ini*, copiez le fichier *php5ts.dll* qui se trouve à la racine de votre répertoire PHP dans le dossier *System* du répertoire Windows (*c:\windows\system* pour Windows 9x/Me ou *c:\winnt\system32* pour Windows NT/2000/XP).

Configuration

Vous allez devoir éditer le fichier *httpd.conf* qui configure le serveur web Apache. Normalement, un lien a été prévu pour cela dans le groupe de programmes Apache créé après l'installation du serveur web (*Configure Apache server* puis *Edit the Apache httpd.conf configuration file*). Si vous ne trouvez pas ce lien, le fichier *httpd.conf* se trouve dans le dossier *conf* du répertoire *Apache*. Avant de vous lancer dans la configuration, assurez-vous que votre serveur web Apache est bien arrêté. Si vous l'avez lancé depuis le lien présent dans le menu **Démarrer**, fermez simplement la console.

Dans ce fichier, recherchez la ligne :

```
#LoadModule unique_id_module modules/mod_unique_id.so
```

Et ajoutez cette ligne juste après :

```
LoadModule php5_module c:/php/php5apache.dll
```

Attention, ne décommentez pas la ligne déjà présente dans le fichier de configuration ; elle a juste été utilisée pour permettre une localisation plus aisée. N'oubliez pas de donner votre propre chemin vers le fichier *php5apache.dll*. Si vous utilisez Apache 2 alors le nom du fichier devient *php5apache2.dll*.

Recherchez ensuite la ligne:

```
#AddModule mod_setenvif.c
```

A sa suite, ajoutez cette ligne:

```
#AddModule mod_php4.c
```

Enfin, recherchez la ligne :

```
#AddType application/x-tar .tgz
```

à la suite de laquelle vous ajoutez cette ligne :

```
AddType application/x-httpd-php .php .php5 .php4
```

Sauvegardez le fichier *httpd.conf* et relancez votre serveur web à l'aide de la console Apache. Vous pouvez tester la bonne prise en compte de PHP en affichant un fichier test en PHP.

Racine
Par défaut, la racine de votre serveur web se trouve dans le dossier htdocs du répertoire Apache. C'est donc là que vous devrez copier vos fichiers .php, .html, etc.

Complément d'installation

Ca y est vous avez installé PHP 5 pour Windows. Vous disposez de la majorité des bibliothèques. Sachez toutefois, que le PHP Group propose également un autre ensemble d'extensions (pour la plupart peu utilisées ou rendues obsolètes) dans un paquetage appelé PECL. Celui-ci est disponible sur le site officiel à l'adresse *http://www.php.net/downloads.php* mais également sur le CDROM fourni.

Pour en profiter, il vous suffit de dézipper son contenu (ou seulement les dll qui vous intéressent) dans le répertoire contenant les extensions PHP dont le chemin est précisé dans le fichier *php.ini* sous le paramètre `extension_dir`. et de les activer en les supprimant les commentaires dans ce même fichier.

Sous Mac OS X

Nous pouvons considérer qu'il existe deux grandes méthodes pour installer PHP dans un environnement Mac OS X. Cela peut ainsi se faire :

- Rien qu'en activant PHP au niveau du serveur préinstallé (si vous disposez de la version serveur de Mac Os X) ;
- En faisant une installation "manuelle" (notamment si vous disposez de la version cliente de Mac Os X ou pour les mises à jours).

Installation par un package

Des packages d'installation préparés pour Mac sont disponibles pour Apache2 et PHP5.

Comme pour PHP4, le site web Entropy.ch met à disposition des internautes un package d'installation de PHP5. Il est bien tenu à jour et suit les remarques des utilisateurs. Vous pouvez le télécharger à cette adresse : http://www.entropy.ch/phpbb2/viewtopic.php?t=1446.

L'installation d'Apache2 pourra se faire directement grâce au package préparé par la Apache Software Foundation : http://www.apple.com/downloads/macosx/unix_open_source/apache.html. Vous n'aurez plus qu'à lancer une rapide compilation pour ensuite profiter d'Apache2.

Activation de PHP sur Mac Os X

Avec OS X, les Mac s'ouvrent beaucoup plus facilement au web, que ce soit dans les versions serveur ou client. Même si les Mac ont toujours fait des serveurs web très sécurisés, du fait, entre autres, de l'absence de ligne de commande, la mise sur pied d'un serveur web sur une machine de la firme de Cuppertino n'était pas, contrairement aux habitudes de la maison, un modèle de simplicité.

Le dernier système d'exploitation d'Apple, Mac OS X, repose sur Darwin, un noyau Unix libre. Avec l'arrivée d'une base Unix, les Mac gagnent en simplicité en matière de web. Alors que les utilisateurs de MacOs n'avaient à leur disposition que quelques serveurs web qu'ils devaient installer eux-mêmes, Mac OS X dispose en natif d'un serveur Apache qui n'attend qu'un claquement de doigts pour être opérationnel. De même, PHP ne demande qu'à être activé. On ne peut toutefois pas se limiter à ses installations par défaut. La volonté légitime d'avoir une installation plus personnalisée (nouveaux modules, optimisation, etc.) passera par une installation ex nihilo d'Apache et de PHP, depuis des sources qu'il faudra compiler. Les mises à jour d'Apache ou du langage PHP passeront par le suivi des annonces faites par Apple.

Ainsi, nous verrons, dans un premier temps, l'activation des services et modules déjà présents à l'installation et, dans un second temps, la compilation et l'installation des programmes depuis un code source.

Activer Apache

Serveur web le plus puissant ou le plus répandu qui soit, peu importe : quand Apple fait quelque chose, il le fait de manière simple et accessible. Les amoureux de la ligne de commande vont en prendre pour leur grade… Lancer Apache sur Mac OS X est encore plus simple que de terminer *Adibou découvre les animaux* en mode facile. Une fois Mac OS X lancé, allez dans le menu **Pomme**, puis dans *Préférences Système*, et sélectionnez *Partage*. Dans la boîte de dialogue qui s'ouvre alors, vous trouverez de nombreuses options relatives à la configuration de votre serveur web Apache. En effet, ce qu'Apple appelle *Partage web* repose, en fait, sur le travail du serveur web (voir fig. 2.21).

Sous le titre *Partage web*, cliquez simplement sur le bouton **Démarrer**. Votre mot de passe vous sera alors demandé ; rentrez-le, puis patientez quelques instants, le temps que le serveur se lance. Voilà, le tour est joué. Si vous obtenez un message d'erreur, il se peut que vous ne soyez pas un utilisateur faisant partie du groupe "`admin`".

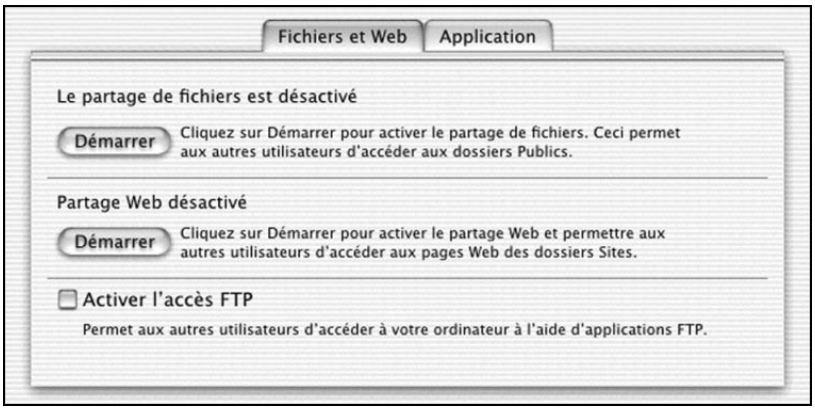


Figure 2.21 : *La boîte de dialogue Partage web sous Mac OS X*

REMARQUE

Le groupe "admin"

Par défaut, le premier utilisateur créé fait partie de ce groupe. Il faut faire partie de ce groupe pour accomplir certaines actions (lancer le partage web par exemple). Pour lancer ou arrêter le serveur Apache, il faudra être soit le premier utilisateur créé, soit un utilisateur qui aura été placé dans le groupe "`admin`*".*

Pour vérifier que tout fonctionne bien, vous pouvez lancer votre navigateur web préféré et vous rendre à l'adresse **http://localhost/**. Vous devriez y trouver une page d'accueil par défaut. Vous pouvez également effectuer cette vérification avec votre nom d'utilisateur, en vous rendant sur votre espace web personnel qui aura été créé automatiquement par le serveur. Si vous êtes l'utilisateur `toto`, votre espace personnel sera alors accessible à l'adresse **http://localhost/toto/**. Cela n'est pas très utile pour le moment, mais cela pourra se révéler intéressant quand il s'agira de faire des tests de scripts. Vous pourrez utiliser votre espace personnel plutôt que le site web de premier niveau (un site en **http://localhost/~toto** plutôt que **http://localhost/**). Les utilisateurs trouveront leur site dans leur répertoire personnel, dans le dossier *Sites*. Pour publier des documents à la racine du site, il faudra placer ces éléments dans le dossier "/*Library*/*Webserver*/*Documents*/".

Activer PHP

Le serveur Apache installé sur Mac OS X comporte déjà le module PHP. Toutefois, celui-ci n'est pas activé par défaut. Il faut donc faire quelques acrobaties accompagnées d'un redémarrage du serveur web pour que toutes les modifications soient prises en compte.

REMARQUE

Faire fausse root

En fouillant sur le web, ou en puisant dans ses connaissances d'unixien, on peut être tenté d'activer le module PHP en passant par l'utilisateur `root`*. Or, ce compte n'existe pas sur Mac OS X. Il est bien sûr possible de le créer. Toutefois, il est fortement recommandé de ne pas créer ce compte. D'une part, on respecte ainsi la philosophie d'Apple vis-à-vis de son système d'exploitation et, d'autre part, l'on garantit une meilleure sécurité au système* `root` *ayant droit de vie et de mort sur le système).*

Pour mener à bien certaines opérations, vous devrez utiliser la commande `sudo` (`su` pour "super-user" et `do` de "to do"). Cette commande vous permet d'avoir accès à des actions normalement réservées à l'utilisateur `root`. Lorsque vous ferez appel à cette commande pour la première fois, le système vous demandera votre mot de passe. Ensuite, vous pourrez l'utiliser directement.

L'activation de PHP consiste principalement en une modification du fichier *httpd.conf*, fichier de configuration du serveur web Apache. Avant de faire des modifications sur ce fichier, il est fortement recommandé d'en faire une copie, afin de garder une porte de sortie en cas de mauvaise manipulation. Rendre ce fichier de configuration inutilisable revient purement et simplement à mettre en berne son site web, Apache ne pouvant alors plus démarrer.

Pour faire cette copie, lancez un terminal (depuis le Finder, dans le dossier *Applications*, puis le dossier *Utilitaires*, choisissez *Terminal*). Une fois le terminal lancé, saisissez cette ligne de commande :

```
sudo cp /etc/httpd/httpd.conf /etc/httpd/httpd.conf.copie
```

Le système vous demande votre mot de passe ; donnez-le lui. Si jamais il vous arrivait malheur lors de l'édition du fichier de configuration, vous n'auriez alors qu'à écraser le fichier de configuration par la sauvegarde que vous aviez faite. Pour cela, utilisez cette commande :

```
sudo cp /etc/httpd/httpd.conf.copie /etc/httpd/httpd.conf
```

Une fois cette opération effectuée, nous pouvons travailler sans danger sur le fichier *httpd.conf* (ou tout au moins sans remords). Nous allons devoir rechercher les références à PHP dans ce fichier, pour nous assurer qu'elles sont bien prises en compte par le serveur lors de son démarrage.

Nous utiliserons l'éditeur de texte Pico pour travailler sur le fichier. Si c'est la première fois que vous utilisez cet éditeur en mode texte, il peut être utile d'en consulter l'aide. Cela vous évitera des heures d'errance, coincé dans un fichier texte qui n'en demandait pas tant.

Puis, une fois Pico lancé, tapez Ctrl + G pour accéder à l'aide intégrée. La plupart des commandes s'obtiennent par le biais de la touche Ctrl suivie d'une lettre. Vous pouvez d'ailleurs voir les fonctions les plus utiles en bas de la fenêtre de terminal. Une fois que vous avez compris les bases de l'utilisation de Pico, vous pouvez lancer l'édition du fichier *http.conf* :

```
sudo pico /etc/httpd/httpd.conf
```

N'oubliez pas que vous devez donner votre mot de passe.

Dans ce fichier, recherchons les références à PHP (Ctrl+W, PHP, Entrée). La première réponse doit être celle-ci :

```
#LoadModule php5_module        libexec/httpd/libphp5.so
```

Le `#` devant la ligne signifie qu'elle est commentée, et que le serveur ne prend pas en compte cette ligne de configuration lors de son démarrage. Supprimez le `#` et refaites une recherche. Vous ne devriez pas avoir à ressaisir "php", Pico gardant en mémoire la dernière occurrence de recherche. Votre nouvelle recherche doit vous amener à cette ligne :

```
#AddModule mod_php4.c
```

De même, supprimez le `#`. Désormais, Apache chargera le module PHP lors de son chargement.

Continuons la recherche. L'occurrence suivante doit se trouver dans ce paragraphe :

```
# AddType allows you to tweak mime.types without actually editing it, or to
# make certain files to be certain types.
#
# For example, the PHP 3.x module (not part of the Apache distribution - see
# http://www.php.net) will typically use:
#
#AddType application/x-httpd-php3 .php3
#AddType application/x-httpd-php3-source .phps
#
# And for PHP 4.x, use:
#
#AddType application/x-httpd-php .php
#AddType application/x-httpd-php-source .phps
```

Ce paragraphe stipule que les fichiers délivrés par le serveur qui ont pour extension *.php3*, *.php* ou *.phps* doivent être traités avec le module PHP. Supprimez aussi les `#` devant ces lignes, sinon le chargement du module PHP ne servirait pas à grand-chose. Attention, ne décommentez pas toutes les lignes, mais seulement les lignes d'instructions, comme ceci :

```
# AddType allows you to tweak mime.types without actually editing it, or to
# make certain files to be certain types.
#
# For example, the PHP 3.x module (not part of the Apache distribution - see
# http://www.php.net) will typically use:
#
 AddType application/x-httpd-php3 .php3
 AddType application/x-httpd-php3-source .phps
#
# And for PHP 4.x, use:
#
 AddType application/x-httpd-php .php
 AddType application/x-httpd-php-source .phps
```

Recherchez maintenant l'occurrence `<IfModule mod_dir.c>`. Vous devez arriver sur un paragraphe comme celui-ci :

```
<IfModule mod_dir.c>
    DirectoryIndex index.html
</IfModule>
```

Ajoutez "`index.php`" à la ligne du milieu, en laissant un espace après "`index.html`". Cela indique au serveur Apache qu'il doit, lorsqu'on lui donne une adresse du type **http://toto.com/titi/**, rechercher un fichier *index.html* dans le dossier */titi*, mais également un fichier *index.php*.

Voilà, toutes les modifications nécessaires ont été effectuées. Vous pouvez sauvegarder le fichier et quitter Pico. Il vous faut maintenant redémarrer le serveur Apache, afin qu'il prenne en compte les modifications que vous avez effectuées dans le fichier de configuration. Faites-le depuis l'interface graphique (*Partage web*, en arrêtant puis démarrant le serveur), ou encore depuis votre terminal en tapant cette commande :

```
sudo apachectl restart
```

Pour vérifier que vos modifications ont bien été intégrées, tapez cette commande dans votre shell :

```
tail /var/log/httpd/error_log
```

Cela aura pour effet d'afficher les traces des messages d'erreur d'Apache. Au début du fichier, vous devez pouvoir trouver cette ligne :

```
[notice] Apache/1.3.23 (Darwin) PHP/4.1.2 configured -- resuming normal operations
```

Une fois que tout est intégré et fonctionnel, profitez de l'occasion pour faire une copie de sauvegarde de votre fichier *httpd.conf*. Cela vous évitera, lors d'une réinstallation, de devoir répéter toutes ces manipulations. Toujours dans votre shell, tapez cette commande :

```
sudo cp /etc/httpd/httpd.conf /etc/httpd/httpd.conf.copie
```


REMARQUE

Une version serveur de pointe
Sur la version serveur de Mac OS X, le serveur de bases de données MySQL fait également partie de l'installation par défaut. Dans la version client, il faut installer soi-même le serveur.

Installation manuelle

Apple fournit des sources prêtes à être compilées qui vous permettront de rester à jour sans avoir à réinstaller tout votre système. Compiler une nouvelle version d'Apache ou du langage PHP peut être obligatoire pour des raisons de sécurité. Ayez en tête que vous devrez alors certainement revoir des options de configuration (tout particulièrement pour *httpd.conf*) pour vos logiciels nouvellement installés. Faites également le tour des modules que vous avez installés et des modules qui seront disponibles et compatibles après la mise à jour. Certains portages peuvent prendre quelque temps. Évitez donc de faire une mise à jour trop rapide, qui vous priverait de modules dont vous avez besoin pour votre site.

Compiler Apache

Avant d'aller plus loin, assurez-vous que vous disposez bien des outils de développement Apple sur votre machine. Si vous n'avez pas le CD qui contient ces logiciels, vous pouvez vous inscrire gratuitement au Developper Network d'Apple. Cet enregistrement vous donnera accès à ces logiciels, que vous n'aurez plus qu'à télécharger et installer. Nous vous recommandons d'ailleurs de faire des mises à jour de ces logiciels. En effet, les versions les plus récentes d'Apache ou de PHP pourraient ne pas se compiler avec des outils trop anciens. Une fois cette installation effectuée, lancez un terminal (depuis le Finder, dans le dossier *Applications*, puis le dossier *Utilitaires*, choisissez *Terminal*). Saisissez alors les lignes de commande suivantes :

```
mkdir install_apache
cd /install_apache
```

Vous créez et entrez dans un répertoire qui vous servira pour cette compilation.

```
wget http://www.apache.org/dist/httpd/httpd_XX.tar.gz
```

Vous téléchargez la dernière version du serveur web Apache directement depuis le site d'Apple.

```
gnutar xzf httpd_XX.tar.gz
```

Vous décompressez le fichier contenant les sources.

```
cd httpd_XX
```

Vous entrez dans le répertoire qui contient les sources.

```
./configure --enable-module=most --enable-shared=max
```

Vous créez le fichier de configuration qui servira lors de la compilation.

```
make
```

Vous compilez les sources.

```
sudo make install
```

Vous installez les exécutables sur votre système.

N'oubliez pas que c'est votre mot de passe qu'il faut donner au système pour qu'il puisse exécuter la commande `sudo`. Dans *httpd_XX.tar.gz*, les *XX* sont à remplacer par le numéro de la version d'Apache concernée. N'oubliez pas d'arrêter puis de relancer votre serveur Apache, comme nous l'avons vu plus haut, afin que ce soit cette nouvelle version qui soit en service.

Compiler PHP

Dans un terminal, rentrez les commandes suivantes :

```
mkdir install_php
cd install_php
```

Vous créez et entrez dans un répertoire qui vous servira pour cette compilation.

```
wget http://www.php.net/distributions/php-XX.tar.gz
```

Vous téléchargez la dernière version du langage PHP directement sur le site de PHP.

```
gnutar -xzf php-XX.tar.gz
```

Vous décompressez le fichier contenant les sources.

```
cd php-XX.tar.gz
```

Vous entrez dans le répertoire qui contient les sources.

```
./configure --with-apxs=/usr/sbin/apxs --disable-pear
```

Sans PEAR c'est mieux ?
Si, comme nous, vous rencontrez des problèmes liés à la bibliothèque `PEAR`, n'hésitez pas à utiliser l'option `--disable-pear`. Vous pouvez toutefois essayer sans, histoire de voir si vous êtes plus chanceux que nous…

Vous créez le fichier de configuration qui servira lors de la compilation.

```
make
```

Vous compilez les sources.

```
sudo make install
```

Vous installez les exécutables sur votre système.

```
sudo cp php.ini-dist /usr/local/lib/php.ini
```

Vous copiez le fichier de configuration de PHP dans un répertoire où Apache saura le trouver.

N'oubliez pas que c'est votre mot de passe qu'il faut donner au système pour qu'il puisse exécuter la commande `sudo`.

Dans *php-XX.tar.gz*, les *XX* sont à remplacer par le numéro de la version du langage PHP concernée.

Assurez-vous enfin que votre fichier *httpd.conf* est bien configuré pour permettre à Apache d'utiliser et d'interpréter correctement PHP.

Quelques liens
En français
Un tutoriel en français très complet et très bien fait pour installer Apache2 et PHP5 :
www.phpmac.com/articles.php?view189
De nombreuses informations pour les développeurs sous Mac Os X (page PHP) :
http://projectomega.online.fr/contents/fr/php/index.php
En anglais
Installer Apache2 et PHP5 sous Mac OS X :
http://laughingmeme.org/archives/001787.html
PHP sur Mac Os X, chez Apple (conseils pour la compilation, modules compatibles, etc.) :
http://developer.apple.com/internet/Mac_OS_X/php.html
Apache et PHP sous Mac Os X par l'un des créateurs de Darwin :
http://www.stepwise.com/Articles/Workbench/2001-10-11.01.html
Apache et PHP sous Mac Os X, par Marc Liyanage :
http://www.entropy.ch/software/macosx/php/

Avec IIS

Installer IIS

L'installation du serveur web IIS est plutôt simple. Insérez le CD-ROM d'installation de Windows NT, 2000 ou XP Pro dans le lecteur de votre machine. Allez dans le menu **Démarrer** puis dans **Paramètres et Panneau de configuration**. Lancez **Ajout/Suppression de programmes**. Cliquez sur l'onglet **Ajouter/Supprimer des composants Windows**, puis cochez la case *Services*

Internet (IIS), validez par OK. Vous pouvez tester la bonne installation d'IIS en ouvrant un navigateur à cette adresse : http://localhost/.

Installer PHP

Vous devrez donc utiliser l'archive zippée disponible sur le site du groupe PHP (*PHP Zip package* dans la section *Windows Binairies* de la section *Download* du site), ou celle fournie sur le CD-ROM. Une fois que vous avez téléchargé le fichier compressé, décompactez-le sur votre disque dur, dans un dossier spécialement créé à cet effet. Sans vouloir vous influencer, *c:\php* serait une bonne idée (évitez tout particulièrement les noms de dossiers avec des espaces). Dans ce dossier, repérez le fichier *php.ini-dist*, renommez-le en *php.ini* , et copiez-le dans votre répertoire Windows (*c:\Windows* ou *c:\WINNT*). Une fois le fichier copié, éditez-le avec votre logiciel préféré.

Recherchez cette ligne :

```
; extension_dir=
```

et renseignez le champ avec l'emplacement où se trouvent les extensions PHP. Par exemple :

```
extension_dir=c:\PHP\extensions\
```

si vous avez installé PHP dans un dossier PHP à la racine de votre disque dur. N'oubliez pas, pour cette ligne et pour toutes les autres, de les décommenter lorsque vous aurez rentré les paramètres nécessaires. Pour cela, il vous suffit de supprimer le `";"` en début de ligne. Attention, ne décommentez que la ligne contenant des paramètres ; laissez les lignes d'explication en commentaire.

Recherchez maintenant la ligne :

```
;browscap=extra/browscap.ini
```

et donnez l'adresse de votre fichier *browscap.ini*. Sous Windows 9x ou Me, vous le trouverez là : *c:\windows\system\inetsrv\browscap.ini* et, sous Windows NT, 2000 ou XP Server, il est situé au bout de ce chemin : *c:\winnt\system32\inetsrv\browscap.ini*. (Cette opération est plus que facultative).

Si vous voulez installer d'autres extensions, il vous suffit de disposer des DLL requises par celles-ci, et de décommenter les lignes les concernant dans le fichier de configuration. Une procédure complète est présentée dans le fichier *install.txt* fourni avec l'archive PHP. Cette procédure est également décrite pour chaque bibliothèque présentée dans ce livre.

Une fois que vous en avez fini avec le fichier *php.ini*, et qu'il est bien placé à la racine de votre dossier Windows, passez à la configuration du serveur lui-même.

Il vous faut travailler dans la console d'administration du serveur IIS. Pour cela, allez dans le **Panneau de configuration**, puis dans **Outils d'administration**. Là, double-cliquez sur **Gestionnaire des services Internet**. Dans la fenêtre qui s'ouvre, faites un clic droit sur le serveur web que vous voulez configurer, et choisissez **Propriétés**. (Attention, déroulez bien l'arborescence pour arriver au site web. Ne travaillez pas sur la machine locale.)

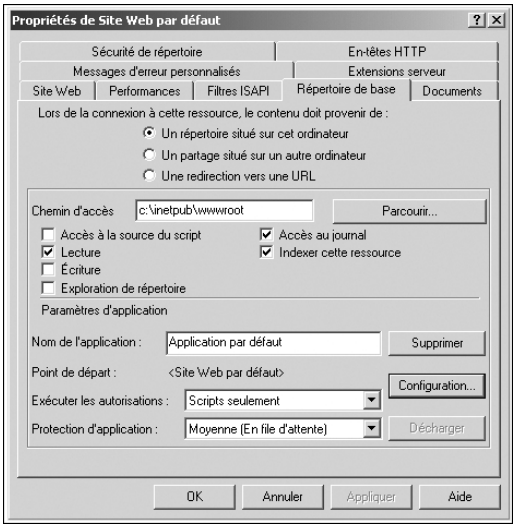


Figure 2.22 : *Configuration de PHP pour IIS*

Cliquez sur l'onglet **Répertoire de base**, puis sur le bouton **Configuration**. Dans la nouvelle fenêtre, cliquez sur *Ajouter*. Dans la nouvelle fenêtre (**Ajout/modification de mappage d'extension de fichier**), cliquez sur *Parcourir*, et donnez le chemin vers *php4isapi.dll* (normalement dans le dossier *sapi* de votre répertoire PHP). Dans le champ extension, donnez *.php*. Vérifiez que la case *Moteur de script* est bien cochée, puis cliquez sur OK pour revenir à la console d'administration. Depuis cette même console, arrêtez et relancer votre serveur. Vous pouvez désormais tester la bonne gestion de PHP. Vous devrez recommencer l'opération pour toutes les extensions que le serveur doit interpréter comme de PHP (*.php3*, *.phtml*, etc.).

Avec iPlanet

Sous Linux

Installer iPlanet

iPlanet s'installe vite et bien. En plus d'être gratuit, il dispose d'une interface web de configuration très puissante, qui pourra aider les plus novices et qui, pour tous, facilite nombre de tâches. Dans bien des situations, cela évite d'avoir à passer par les fichiers de configuration. Nous présentons ici une installation pour GNU/Linux sur une distribution RedHat. Pour une installation sur un autre système de type UNIX, reportez-vous aux liens présentés en fin de partie.

Pour installer iPlanet, à moins que vous ne disposiez de votre propre copie, téléchargez-en une version sur le site de Sun (wwws.sun.com/software/download/download/5126.html). Le serveur est distribué sans contrepartie ; il vous suffit de laisser quelques informations à Sun. Pendant que le téléchargement suit son cours, vous allez pouvoir faire quelques petites manipulations préparatoires. La version de iPlanet distribuée par Sun pour Linux a été préparée pour une RedHat 6.0. Les autres distributions (hors Mandrake) auront certainement quelques difficultés à s'adapter. Les versions postérieures à la 6.0 posent d'ailleurs un problème assez gênant, puisqu'elles interrompent carrément le script d'installation du serveur. En fait, iPlanet demande

la librairie `ncurses` dans sa quatrième version, alors qu'aujourd'hui, c'est la version 5.2 qui est fournie avec les systèmes RedHat. Il est possible de tricher en faisant un lien symbolique de la version installée vers une prétendue version 4 ; version qui donnera alors entière satisfaction au script d'installation. Pour cela, en étant logué en tant que `root`, rentrez la commande suivante :

```
ln -s /usr/lib/libncurses.so.5.2 /usr/lib/libncurses.so.4
```

Pour vérifier quelle est la version de `ncurses` installée sur votre système, tapez la commande :

```
locate ncurses
```

Notez alors le chemin qui s'affiche, et utilisez-le dans la commande donnée plus haut.

Cette manœuvre ne met absolument pas en péril le système d'exploitation, pas plus qu'elle ne compromet le bon fonctionnement du serveur. La librairie concernée n'est en effet utilisée que lors de la création de l'interface d'administration. Lors de nos tests, cette procédure a parfaitement fonctionné, n'entraînant aucun dysfonctionnement du serveur, du système Linux, ni de l'interface graphique elle-même. Une fois le lien créé et le téléchargement terminé, vous pouvez passer à l'installation du serveur.

Avant de vous lancer dans la procédure d'installation, assurez-vous que vous disposez bien de tous les outils logiciels nécessaires : compilateur, gestionnaire de fichiers compressés, etc. Les utilisateurs de Sun trouveront tout le nécessaire sur le site **http://www.sunfreeware.com**.

Rendez-vous dans le dossier contenant l'archive iPlanet et décompactez-la. Entrez dans le dossier *iws* qui vient alors d'être créé. En tant qu'utilisateur `root` (commande `su` pour passer en `root`), lancez le script d'installation (`./setup`). Renseignez ensuite tous les champs demandés et répondez aux questions qui restent assez classiques (dossier racine du site, création d'un utilisateur et d'un groupe pour le serveur, etc.). Vous pouvez d'ailleurs choisir entre une configuration express ou une configuration avancée. À vous de voir ce qui convient le mieux à l'installation que vous êtes en train d'effectuer. Si vous ne destinez pas le serveur à une mise en exploitation, autant aller au plus vite. Retenez bien l'identifiant (*login*) et le mot de passe qui vous sont demandés pour l'accès à l'interface de gestion.

Une fois iPlanet installé, passons à la compilation de PHP.

Compiler PHP

Tout d'abord, téléchargez la dernière version de PHP sur le site officiel (**http://www.php.net**), ou bien utilisez celle fournie sur le CD-ROM. Passez en utilisateur `root`, et vérifiez que les paramètres suivants sont bien situés dans votre `PATH` (commande `echo $PATH`). Si elle n'y sont pas, ajoutez-les :

```
:/usr/local/bin:/usr/sbin:/usr/bin:/usr/ccs/bin
```

Décompactez l'archive :

```
# tar xzvf php-XX.tar.gz
```

Rentrez dans le répertoire créé lors du décompactage, puis rentrez cette commande :

```
# ./configure --with-nsapi=/opt/netscape/suitespot/ --enable-libgcc
```

Le chemin suivant le paramètre `--with-nsapi` varie selon votre installation. À vous de donner le bon chemin. Sous Linux, ce doit être quelque chose comme : */usr/iplanet/servers/*.

Si le serveur de base de données MySQL est installé sur votre machine, vous pouvez également donner ce paramètre comme option de configuration `--with-mysql=/usr/lib/mysql`.

Une fois la configuration effectuée, rentrez alors la commande :

```
# make
```

puis, finalement :

```
# make install
```

PHP est maintenant installé sur votre système. Vous pouvez passer à la configuration du serveur web.

Configuration d'iPlanet

La configuration du serveur iPlanet se résume à l'édition de quelques fichiers de configuration ponctuée d'un redémarrage dudit serveur. Rendez-vous dans le répertoire contenant les fichiers de configuration de votre serveur (quelque chose comme */usr/iplanet/servers/https-localhost.localdomain/config/*). Faites attention, ne vous trompez pas entre le dossier *https-loalhost.localdomain* et le dossier *https-admserv*. Le premier est le bon ; c'est celui qui contient les informations et documents relatifs à votre serveur web public. Le second ne concerne que le serveur utilisé par l'interface web de configuration, Sun ayant offert à l'interface web son propre serveur (ce qui est plutôt une bonne idée quand on sait ce que peut engendrer une mauvaise manipulation des fichiers de configuration). N'ayez d'ailleurs pas peur lors de vos exercices de configuration : il existe déjà sur votre machine un répertoire de sauvegarde qui contient une copie des fichiers de configuration du serveur web. Situé au même niveau que le dossier *config*, le dossier *config-bk* pourra vous sortir de l'ornière en cas de pépin. Il vous suffira de copier tout simplement son contenu dans le répertoire *config*.

Commencez d'abord par éditer le fichier *mime.types* pour y ajouter la ligne suivante :

```
type=magnus-internal/x-httpd-php        exts=php,php3,php4,php5,phtml
```

Sauvegardez *mime.types*, puis passez à *magnus.conf*. Ajoutez les lignes suivantes à la fin de la section des inits, en début de fichier :

```
Init fn="load-modules"
funcs="php5_init,php5_close,php5_execute,php5_auth_trans"
shlib="/php5/nsapiPHP5.dll"
Init fn="php5_init" errorString="L'initialisation du PHP a échoué !"
✂ LateInit="yes"
```

Le chemin utilisé pour shlib varie en fonction du système d'exploitation. Pour Linux, ce sera quelque chose comme */opt/netscape/suitespot/bin/libphp5.so* ou */usr/iplanet/servers/bin/libphp5.so*. Donnez simplement le chemin qui pointe vers la librairie `PHP 5`, précédemment installée lors de la compilation. Une fois *magnus.conf* complété, sauvegardez-le, puis ouvrez *obj.conf*. En suivant le modèle et l'aspect général du fichier, ajoutez-y le bloc de texte suivant (un nouvel objet) :

```
</Object>
<Object name="x-httpd-php">
ObjectType fn="force-type" type="magnus-internal/x-httpd-php"
Service fn=php5_execute
```

```
</Object>
```

Ensuite, dans l'objet *default*, vous devrez ajouter la ligne suivante :

```
Service fn="php5_execute" type="magnus-internal/x-httpd-php"
```

et ce avant chaque ligne `AddLog` et après chaque ligne `ObjectType`. Voici un exemple, pris pour une configuration sous Linux :

```
<Object name=default>
NameTrans fn="NSServletNameTrans" name="servlet"
NameTrans fn="pfx2dir" from="/servlet"
dir="/usr/iplanet/servers/docs/servlet" name="ServletByExt"
NameTrans fn=pfx2dir from=/mc-icons dir="/usr/iplanet/servers/ns-icons"
name="es-internal"
NameTrans fn="pfx2dir" from="/manual"
dir="/usr/iplanet/servers/manual/https" name="es-internal"
NameTrans fn=document-root root="$docroot"
PathCheck fn=unix-uri-clean
PathCheck fn="check-acl" acl="default"
PathCheck fn=find-pathinfo
PathCheck fn=find-index index-names="index.html,home.html"
ObjectType fn=type-by-extension
Service fn="php5_execute" type="magnus-internal/x-httpd-php"
ObjectType fn=force-type type=text/plain
Service fn="php5_execute" type="magnus-internal/x-httpd-php"
Service type="magnus-internal/jsp" fn="NSServletService"
Service method=(GET|HEAD) type=magnus-internal/imagemap fn=imagemap
Service method=(GET|HEAD) type=magnus-internal/directory fn=index-common
Service method=(GET|HEAD|POST) type=*~magnus-internal/* fn=send-file
Service fn="php4_execute" type="magnus-internal/x-htt
AddLog fn=flex-log name="access"
</Object>
```

Une fois le fichier *obj.conf* correctement renseigné, sauvegardez-le et relancez le serveur web afin qu'il prenne en compte les dernières modifications (commande `./restart`). Vous pouvez alors tester la bonne prise en compte de PHP avec une simple page placée à la racine de votre serveur web.

Sous Windows

L'installation et la configuration du serveur web iPlanet sous Windows ne présentent pas de difficulté particulière. Bien que nombreuses, les étapes nécessaires ne sont en rien insurmontables.

Installation du serveur web iPlanet

Tout d'abord, téléchargez directement la version Windows du serveur sur le site de Sun (**wwws.sun.com/software/download/download/0104.html**). Lancez le programme d'installation. La procédure vous est proposée en trois déclinaisons : express, typique ou personnalisée. Si c'est la première fois que vous installez iPlanet sur votre machine, choisissez l'option typique, qui est largement

suffisante, et qui vous permettra même de modifier certains paramètres assez pratiques. Premier point important, vous allez devoir choisir un dossier pour l'installation du serveur web. Conservez le chemin proposé par le programme d'installation (évitez le dossier *Program Files* qui serait épineux à utiliser dans des fichiers de configuration). Donnez ensuite un nom d'utilisateur et un mot de passe, qui seront utilisés pour l'interface d'administration web. Conservez précieusement ces informations, sans quoi vous seriez un peu gêné pour gérer votre serveur. Le port d'administration (par défaut 8888) ainsi que le port web (80) n'ont pas à être modifiés, à moins que vous n'ayez une idée bien spécifique. Étape suivante, spécifiez et notez bien le dossier racine du site. Les dernières étapes de préparation de l'installation concernent l'utilisation éventuelle d'un annuaire LDAP ou de Java. Lorsque vous aurez répondu à toutes ces questions, l'installation proprement dite pourra débuter. Une fois iPlanet correctement mis en place, vous pouvez tester son bon fonctionnement en rentrant l'adresse **http://localhost/** dans un navigateur web de votre choix.

Installer PHP

Téléchargez la dernière version de PHP pour Windows zippée sur le site officiel du groupe PHP (**http://www.php.net**), ou bien utilisez la version disponible sur le CD-ROM. Décompactez cette archive dans un dossier *c:\php*, ou dans tout autre dossier simple, facile à reprendre pour les éditions de fichiers de configuration à venir. Copiez le fichier *php4ts.dll* ainsi décompacté dans votre dossier Windows (*c:\WINNT*). Toujours dans le dossier qui a servi pour le décompactage, trouvez le fichier *php.ini-dist*, et renommez-le en *php.ini*. Le changement de nom effectué, éditez ce même fichier, soit avec votre éditeur de texte préféré, soit avec la ligne de commande suivante dans une fenêtre MSDos :

```
edit php.ini
```

Dans ce fichier, recherchez la ligne :

```
extension_dir = ./
```

et renseignez-la avec l'emplacement sur votre système du dossier contenant les extensions PHP. Si vous avez décompacté PHP dans un répertoire *c:\php*, vous devez remplir la ligne ainsi :

```
extension_dir= c:\php\extensions\
```

Sauvegardez le fichier, puis copiez-le à la racine de votre répertoire *WINNT* :

```
copy php.ini c:\WINNT\
```

Il vous faut maintenant créer une association de fichiers pour PHP. Dans une fenêtre MSDos, rentrez les deux lignes de commande suivantes, en validant par la touche Entrée à chaque fin de ligne :

```
assoc .php=PHPScript
ftype PHPScript=c:\php\php.exe %1 %*
```

À noter que l'adresse donnée dans la seconde ligne est à modifier selon vos propres paramètres.

Vous pouvez passer maintenant à la configuration du serveur web iPlanet pour qu'il prenne en compte PHP.

Configurer PHP

Tout d'abord, stoppez votre serveur (depuis l'interface web de configuration, dans la rubrique *on/off*).

Commencez d'abord par éditer le fichier *mime.types* pour y ajouter la ligne suivante :

```
type=magnus-internal/x-httpd-php        exts=php,php3,php4,phtml
```

Sauvegardez *mime.types*, puis passez à *obj.conf*. Ajoutez les lignes suivantes à la fin de la section des inits, en début de fichier :

```
Init fn="load-modules"
funcs="php5_init,php5_close,php4_execute,php5_auth_trans"
shlib="c:/php/sapi/php5nsapi.dll"
Init fn="php5_init" errorString="L'initialisation du PHP a échoué !"
✂ LateInit="yes"
```

Le chemin utilisé pour shlib varie en fonction de votre installation. Avec l'exemple cité plus haut, il faudrait donner ce chemin : *c:/php/sapi/php4nsapi.dll*. En suivant le modèle et l'aspect général du fichier, ajoutez-y le bloc de texte suivant (un nouvel objet) :

```
</Object>
<Object name="x-httpd-php">
ObjectType fn="force-type" type="magnus-internal/x-httpd-php"
Service fn=php5_execute
</Object>
```

Ensuite, dans l'objet *default*, vous devrez ajouter la ligne suivante :

```
Service fn="php5_execute" type="magnus-internal/x-httpd-php"
```

et ce avant chaque ligne `AddLog` et après chaque ligne `ObjectType` (pour un exemple de fichier de configuration, voir la partie *Configuration sous Unix*).

Une fois ces fichiers édités et sauvegardés, vous pouvez relancer le serveur web et tester votre nouvelle configuration.

Autres

Installation sous les autres systèmes d'exploitation
Installation sous HP-UX :
www.php.net/manual/fr/install.hpux.php
Installation sous Solaris :
www.php.net/manual/fr/install.solaris.php
Installation sous Unix-OpenBSD :
www.php.net/manual/fr/install.openbsd.php
Tutoriel d'installation pour PHP sous Windows NT :
http://benoit.noss.free.fr/php/install-php4.html

2.2. Le fichier de configuration php.ini

PHP se configure en deux fois. Dans un premier temps, lors de sa compilation, pour assurer le support de certaines bases de données, d'un serveur web sous la forme d'un module pour activer certaines fonctions, etc. ; et, dans un second temps, après son installation, par l'édition et le renseignement du fichier *php.ini*.

Le fichier *php.ini* contrôle bon nombre des comportements de PHP. Pour que PHP puisse lire et comprendre ce fichier, il doit impérativement avoir pour nom *php.ini*. Au démarrage, PHP cherche ce fichier dans son répertoire de travail, dans le chemin désigné par la variable d'environnement `PHPRC` puis, enfin, dans le chemin défini lors de la compilation. Sous Windows, le chemin correspond au répertoire *Windows* ; sous Linux, par défaut, il s'agit du répertoire */usr/local/lib*.

Si PHP a été compilé en module, le fichier *php.ini* ne sera lu qu'une seule fois, lors du lancement du serveur web. Si PHP fonctionne en CGI, *php.ini* sera lu à chaque utilisation de PHP.

Si vous utilisez des constantes dans vos valeurs et que ces constantes appartiennent à une extension chargée dynamiquement (extension PHP ou Zend), vous ne pouvez utiliser ces constantes qu'après la ligne de configuration qui charge cette extension.

Les valeurs utilisées dans le fichier *php.ini-dist* correspondent aux valeurs par défaut de PHP. Cela veut dire que, si vous n'utilisez pas de *php.ini* ou que vous effacez les lignes qui suivent, les valeurs et paramètres utilisés par PHP seront identiques à ceux présentés dans *php.ini-dist*.

Pour créer un fichier *php.ini*, utilisez le fichier *php.ini-dist* présent dans votre répertoire PHP. Ce fichier contient un squelette de configuration, qu'il vous appartiendra d'adapter à votre système selon vos besoins. Notez qu'il existe également un fichier *php.ini-recommended*, que vous trouverez, lui aussi, à la racine de votre dossier PHP. Ce fichier est une recommandation de configuration pensée pour vous par les développeurs de PHP. Comme le précise l'en-tête du fichier, cette configuration peut rendre PHP incompatible avec certaines applications. Elle rend également le développement plus difficile et plus rigoureux. Une bonne solution consiste à se baser sur ce fichier, en passant en revue toutes les variables pour vérifier si elles conviennent à votre environnement de travail. Vous ne modifierez ainsi que le strict nécessaire, tout en conservant une configuration plus sûre et plus performante que celle offerte par défaut par la simple mise en place du *php.ini-dist*.

Dans cette partie, nous allons passer en revue toutes les options présentes dans le fichier *php.ini-dist* ; fichier qui, une fois renseigné, deviendra votre propre *php.ini*.

La syntaxe du fichier *php.ini* est simple. Toute ligne commençant par un point-virgule ou toute chaîne de caractères contenue entre crochets ne sera pas interprétée par PHP.

Les directives utilisent une syntaxe simple :

```
directive = valeur
```

Attention, les noms des directives sont sensibles à la casse : respectez donc scrupuleusement l'utilisation des majuscules ou minuscules (EMMA est différent de Emma).

La valeur peut consister en une chaîne de caractères, un nombre, une constante PHP (ex. : `E_ALL` ou `M_PI`), une des constantes INI (`On`, `Off`, `True`, `False`, `Yes`, `No` et `None`), une expression (ex. : `E_ALL & ~E_NOTICE`), ou, enfin, une chaîne de caractères entre guillemets (`"emma"`).

Dans le fichier *php.ini*, les expressions sont limitées aux opérateurs binaires et aux parenthèses :

```
|        binaire OR
&        binaire AND
~        binaire NOT
!        booleen NOT
```

Les opérateurs booléens peuvent être activés en utilisant les valeurs `1`, `On`, `True` ou `Yes`. Pour les désactiver, utilisez les valeurs `0`, `Off`, `False` ou `No`.

Une chaîne vide peut être indiquée soit en n'écrivant rien après le `=`, soit en utilisant le mot-clé `none`.

Attention, dans l'expression `toto = "none"`, la chaîne de caractères ne sera pas vide, mais contiendra le mot `none` ; une chaîne vide serait `toto = none`.

Options PHP de base

Balises

`short_open_tag = On`

permet d'utiliser le tag court d'ouverture de script PHP `< ?` (plutôt que `< ?php` ou `<script>`).

`asp_tags = Off`

permet d'utiliser les tags ASP `<% %>`.

Serveur

`engine = On`

autorise le moteur de langage de script PHP sous Apache.

`zend.ze1_compatibility_mode = Off`

active le module de compatibilité avec le Zend Engine 1 (utilisé dans PHP4.x).

`expose_php = On`

permet d'indiquer que PHP est installé sur le serveur, et que, par conséquent, on peut l'utiliser. Cela ne constitue pas à proprement parler un problème de sécurité, mais cela signale tout de même aux utilisateurs que PHP est installé sur le serveur.

`y2k_compliance = Off`

Compatibilité an 2000. Il est déconseillé d'activer cette option, car elle pourrait poser des problèmes avec des navigateurs non compatibles an 2000.

Limites des ressources

`max_execution_time = 30`

Indiquez le temps maximum d'exécution des scripts, en secondes (ici : 30 secondes).

`memory_limit = 8M`

Indiquez la quantité maximale de ressource mémoire utilisable par les scripts (ici : 8 Mb).

Sécurité

Mode sécurisé

`safe_mode = Off`

Activation ou désactivation du mode sécurisé.

`safe_mode_gid = Off`

À l'ouverture des fichiers, le mode sécurisé utilise, par défaut, des vérifications comparatives des UID. Pour passer à des comparaisons sur le GID, activez cette commande.

`safe_mode_include_dir =`

Si `safe_mode` est activé, les vérifications des UID/GID sont ignorées lorsqu'un fichier contient des inclusions de fichiers provenant de ce répertoire ou de ses sous-répertoires. Le chemin concerné doit être présent dans le `include_path`, ou alors c'est le chemin complet qui doit être donné pour l'inclusion.

`safe_mode_exec_dir =`

Si le mode sécurisé est activé, seuls les exécutables qui se situent dans le répertoire renseigné ici seront autorisés à l'exécution via la famille des fonctions exécutables (comme `exec()`).

`open_basedir =`

Indiquez le chemin du répertoire sur lequel vous souhaitez limiter les opérations sur les fichiers. Ceci sera valable pour ce répertoire et ses sous-répertoires.

`safe_mode_allowed_env_vars = PHP_`

La configuration de certaines variables d'environnement peut présenter des failles de sécurité. Cette directive comprend une liste de préfixes séparés par des virgules. En mode sécurisé, l'utilisateur peut uniquement modifier les variables d'environnement dont le nom commence par les préfixes indiqués ici. Par défaut, les utilisateurs ne pourront configurer que les variables d'environnement commençant par `PHP_` (ex. : `PHP_FOO=BAR`). Si cette directive est vide, l'utilisateur pourra modifier toutes les variables d'environnement !

Autres

```
safe_mode_protected_env_vars = LD_LIBRARY_PATH
```

Indiquez, en les séparant par une virgule, les variables d'environnement que l'utilisateur final ne pourra pas modifier en utilisant `putenv()`. Ces variables seront protégées, même si la directive `safe_mode_allowed_env_vars` est configurée pour que l'on puisse les changer.

```
disable_functions =
```

Indiquez ici, séparées par des virgules, les fonctions que vous souhaitez interdire pour des raisons de sécurité. L'activation ou non du mode sécurisé n'a aucune influence sur cette directive.

```
enable_dl = On
```

Activation ou non de la fonction `dl()` qui permet de charger des DLL à la volée. Cette fonction ne marche pas correctement sur des serveurs multi-tâches (*multithread*) comme, par exemple, IIS ou Zeus, et est automatiquement désactivée sur ces derniers.

```
cgi.force_redirect = 1
```

`cgi.force_redirect` est nécessaire pour apporter de la sécurité si l'on fait tourner PHP en tant que CGI sous la plupart des serveurs web. Si vous laissez cette directive vide, PHP l'interprétera comme activée par défaut. Il est tout à fait déconseillé de désactiver cette directive (hormis sous IIS, pour lequel il vaut mieux l'activer).

```
cgi.redirect_status_env =
```

Si `cgi.force_redirect` est activé, et que vous ne vous trouvez pas sous un serveur Apache ou Netscape (iPlanet), vous devrez déterminer un nom de variable d'environnement que PHP interrogera afin de savoir s'il peut ou non continuer l'exécution. Ayez une idée précise de ce que vous souhaitez faire avant de déterminer tout cela, la configuration de cette variable pouvant provoquer des problèmes de sécurité !

```
cgi.fix_pathinfo=0
```

Si `cgi.fix_pathinfo` est activé , PHP CGI corrigera son chemin pour se conformer aux spécifications. Si vous n'activez pas ce paramètre (en le laissant à 0), PHP se comportera comme dans les versions antérieures. Il est conseillé d'utiliser `SCRIPT FILENAME` plutôt que `PATH TRANSLATION`.

```
enable_dl = On
```

Gestion des erreurs et récupération des messages d'erreur

`error_reporting` est un champ de bits. Vous pouvez déterminer quel type et quel niveau de rapport d'erreur vous souhaitez récupérer.

```
E_ALL
```

Toutes les erreurs et avertissements.

`E_ERROR`

Erreurs fatales du programme.

`E_WARNING`

Avertissements du programme (qui ne sont pas des erreurs fatales).

`E_PARSE`

Erreurs d'analyse lexicale à la compilation.

`E_NOTICE`

Indications du programme. Ce sont des avertissements qui résultent généralement d'un bug dans votre code, mais il se peut que ce soit intentionnel. C'est le cas, par exemple, si l'on utilise une variable non initialisée en considérant qu'elle sera automatiquement initialisée par une chaîne vide.

`E_STRICT`

Permet à PHP de vous indiquer des modifications dans votre code qui assureraient une meilleure inter-opérabilité et une meilleure compatibilité ascendante.

`E_CORE_ERROR`

Erreurs fatales survenant lors du démarrage initial de PHP.

`E_CORE_WARNING`

Avertissements (qui ne sont pas des erreurs fatales) survenant lors du démarrage initial de PHP.

`E_COMPILE_ERROR`

Erreur fatale à la compilation.

`E_COMPILE_WARNING`

Avertissement d'erreur à la compilation (qui ne sont pas des erreurs fatales).

`E_USER_ERROR`

Message d'erreur provoqué par l'utilisateur.

`E_USER_WARNING`

Message d'avertissement provoqué par l'utilisateur.

`E_USER_NOTICE`

Message d'indication provoqué par l'utilisateur.

Exemples :

```
error_reporting = E_ALL & ~E_NOTICE & ~E_STRICT
```

Montrer toutes les erreurs, sauf les messages d'indication et les messages de qualité de code.

```
error_reporting = E_COMPILE_ERROR|E_ERROR|E_CORE_ERROR
```

Ne montrer que les erreurs.

```
display_errors = On
```

Active l'affichage des erreurs. Pour un site de production, il est conseillé de désactiver cette fonction et d'utiliser, à sa place, `error logging`. Si vous gardez `display_errors` activée, vous risquez de révéler des informations de sécurité aux utilisateurs finals comme, par exemple, des chemins d'accès de votre serveur web, le schéma de votre base de données, ou d'autres informations encore.

```
display_startup_errors = Off
```

Même si `display_errors` est activé, les erreurs qui se produisent lors du démarrage de PHP ne sont pas affichées. Il est fortement conseillé de ne pas activer cette fonction, sauf pour du débogage.

```
log_errors = Off
```

Permet de stocker les erreurs dans un fichier de logs. S'il s'agit d'un site de production, il est fortement conseillé d'utiliser cette directive au lieu de `error_display`.

```
track_errors = Off
```

Archive le dernier message d'erreur ou d'avertissement dans `$php_errormsg`.

```
html_errors = Off
```

Interdit l'inclusion de tags HTML dans les messages d'erreur. N'utilisez pas cette fonction sur des serveurs de production.

```
error_prepend_string = "<font color=ff0000>"
```

Chaîne à intégrer avant un message d'erreur.

```
error_append_string = "</font>"
```

Chaîne à intégrer après un message d'erreur.

```
error_log = filename
```

Placer les traces d'erreur dans un fichier particulier. Remplacez `filename` par le nom du fichier que vous voulez utiliser.

```
error_log = syslog
```

Placer les traces d'erreur dans *syslog* (Event Log sur Windows NT, non valide sous Windows 95).

```
define_syslog_variables  = Off
```

Cette directive active ou désactive la définition de diverses variables `syslog` comme, par exemple, les variables `$LOG_PID`, `$LOG_CRON`, etc. Il est conseillé de désactiver cette directive

dans un souci de performance. Vous pourrez définir ces variables dans le programme en faisant appel à la fonction `define_syslog_variables()`.

```
warn_plus_overloading = Off
```

Prévenir si l'opérateur + est utilisé dans les chaînes.

Gestion des fichiers

Include

```
include_path =
```

Indique quels répertoires doivent être explorés lorsqu'un fichier est inclus.

Sous UNIX, il devra être donné de cette façon : `include_path = ".:/php/includes"` et, sous Windows, de la façon suivante : `include_path = "c:\php\includes"`. Vous pouvez en spécifier plusieurs, en les séparant par un deux points (`:`) sous Linux, et par un point-vigule (`;`) sous Windows, comme ceci : `include_path = "c:\php\includes;c:\php\emma \includes"`.

Ouverture distante de fichiers

```
allow_url_fopen = On
```

Active ou désactive l'autorisation de traitement des URL (comme `http://` ou `ftp://`) en tant que fichiers.

```
from="john@doe.com"
```

Définit le mot de passe anonyme du FTP. Ici, une adresse e-mail sera demandée.

Publication de fichiers

```
file_uploads = On
```

Autorise (ou non) l'upload de fichiers HTTP.

```
upload_tmp_dir =
```

Indiquez, si besoin, le répertoire temporaire où vous souhaitez archiver les fichiers HTTP publiés (uploadés). Si vous ne remplissez pas ce champ, ils seront placés dans le répertoire temporaire par défaut du système.

```
upload_max_filesize = 2M
```

Taille maximale autorisée des fichiers en upload.

Gestion des données

Depuis PHP 4.0.3, `track_vars` est toujours activé.

```
arg_separator.output = "&amp;"
```

Le séparateur des arguments utilisé dans les URL générées par PHP est, par défaut, '`&`'.

```
arg_separator.input = ";&"
```

Liste des caractères que doit détecter PHP pour séparer les arguments dans les URL (par défaut les caractères '`&`' et '`;`'). Chaque caractère est considéré, dans cette directive, comme un séparateur.

```
variables_order = "EGPCS"
```

Cette directive décrit l'ordre dans lequel PHP enregistre les variables passées par les méthodes `GET`, `POST` ou `Cookie`, les variables d'environnement et les variables incluses (respectivement G, P, C, E et S). Cet enregistrement prend en compte les valeurs de gauche à droite, les nouvelles valeurs prenant le pas sur les plus anciennes.

```
register_globals = Off
```

Cette directive vous permet de choisir si vous enregistrez les variables EGPCS comme des variables globales ou pas. Par mesure de sécurité, il est préférable de laisser ce paramètre à `Off`.

```
register_long_arrays = On
```

Cette directive vous permet de continuer à utiliser la méthode `HTTP_GET_VAR` . Si vous n'utilisez pas cette méthode (en préférant la méthode `$_GET["Varaible"]`), il est préférable de ne pas activer la directive pour optimiser les performances.

```
register_argc_argv = On
```

Cette directive définit si PHP déclare les variables `argv` et `argc` (qui pourraient contenir les informations `GET`). Si vous n'utilisez pas ces variables, vous pouvez désactiver cette directive pour améliorer les performances.

```
post_max_size = 8M
```

Taille maximale des données `POST` que PHP acceptera.

Les magic quotes

Les magic quotes sont des directives qui vous permettent d'échapper les caractères qui pourraient provoquer des erreurs dans votre code. Par exemple, une apostrophe, un slash, etc. dans une variable issue d'un formulaire qui pourraient être interprétés comme une fin de chaîne ou autre caractère interprété par PHP :

```
<form action="recupere.php">
<input type="hidden" name="exemple" value="L'aventure c'est l'aventure !">
<input type="submit">
</form>
```

Puis, à l'aide de ce script :

```
<?php
echo $_GET["exemple"];
?>
```

vous récupérez les données contenues dans *recupere.php*. Si les magic quotes sont activées, vous obtenez ceci :

L\'aventure c\'est l\'aventure !

Sans les magic quotes, nous avons ceci :

```
<?php
echo $_GET["exemple"];
?>
```

L'aventure c'est l'aventure !

`magic_quotes_gpc = On`

Active les magic quotes pour les données entrantes `GET`, `POST` et `Cookie`.

`magic_quotes_runtime = Off`

Désactive les magic quotes pour les données générées par le programme (ex. : données SQL, données provenant des fonctions `exec()`, etc.).

`magic_quotes_sybase = Off`

Si cette directive est activée, des apostrophes seront utilisées en lieu et place des "backslashes" pour les magic quotes (style `Sybase`, c'est-à-dire `''` au lieu de `\'`).

Sessions

`session.save_handler = files`

Il sert à redéfinir la manière dont seront gérées les sessions. Par défaut, les sessions sont enregistrées dans des fichiers. C'est grâce à ces fichiers que PHP retrouvera les données d'une session. Si vous souhaitez une gestion personnalisée des sessions indiquez alors "`user`" (voir chapitre sur les sessions).

`session.save_path = /tmp`

Répertoire par défaut où stocker les sessions. Il vaut mieux éviter que ce répertoire soit lisible par tous, auquel cas la sécurité du site pourrait être compromise. Quelqu'un pourrait usurper l'identificateur d'un autre. Par défaut, il vaut `/tmp`. Il faut donc impérativement modifier ce paramètre pour pouvoir faire fonctionner les cookies sous Windows.

`session.use_cookies = 1`

Permet d'indiquer si le gestionnaire de sessions doit utiliser un cookie chez le client pour mémoriser l'identifiant de session. Par défaut l'utilisation de cookies est activée (valeur à 1).

```
session.name = PHPSESSID
```

Nom de l'identifiant de la session. Il ne contient que des caractères alphanumériques.

```
session.auto_start = 0
```

Indique si une session doit automatiquement être lancée lors de la première requête. Cette option est désactivée par défaut (valeur à 0).

```
session.cookie_lifetime = 0
```

Permet de définir la durée de vie du cookie de session (en secondes). La valeur 0 (par défaut) signifie que le cookie doit être effacé à la fermeture du navigateur.

```
session.cookie_path = /
```

Permet de définir le chemin à utiliser par les cookies. Par défaut, il est mis à `/`.

```
session.cookie_domain =
```

Permet de définir le domaine sur lequel est restreint le cookie.

```
session.serialize_handler = php
```

Permet de définir le gestionnaire de la sérialisation/désérialisation. Les formats PHP et WDDX sont supportés, WDDX n'étant disponible que si PHP a été compilé avec WDDX.

```
session.gc_probability = 1
```

Permet de définir la probabilité d'exécution de `gc` (garbage collector) à chaque requête. `gc` est le dispositif permettant de supprimer les sessions qui ne sont plus valides. Généralement, les sessions sont enregistrées sous forme de fichiers, et `gc` se charge de supprimer les fichiers périmés. Par défaut, `gc` est requis à chaque appel (la valeur vaut 1).

```
session.gc_maxlifetime = 1440
```

Après ce laps de temps (en secondes), une donnée sera considérée comme périmée et pouvant être supprimée par `gc`.

```
session.referer_check =
```

Permet de ne valider une session que si l'utilisateur provient d'un site en particulier (typiquement le vôtre). Il suffit de renseigner tout ou partie du nom de domaine. Si cette chaîne de caractères est contenue dans le `HTTP_REFERER` fourni par le navigateur, ou si cette valeur retournée est vide, alors la session est validée. Par défaut, cette chaîne de caractères est vide.

```
session.entropy_file =
```

Permet de définir le chemin vers un fichier externe qui sera utilisé pour générer (de façon pseudo-aléatoire) un identifiant de session. Dans le monde UNIX, il est possible de définir `/dev/random` ou encore `/dev/urandom`.

```
session.entropy_length = 0
```

Permet de déterminer le nombre d'octets lus dans le fichier déclaré par `session.entropy_file`. Par défaut, celui-ci étant désactivé, il vaut 0.

```
session.cache_limiter = nocache
```

Permet de définir l'en-tête de contrôle du cache qui doit être envoyé avec chaque script généré à partir de variables de sessions. Il peut être défini comme `none`, `nocache`, `private`, `private_no_exprire` ou `public`.

```
session.cache_expire = 180
```

Permet de définir l'en-tête définissant la durée de validité (en minutes) d'une page générée à partir de variables de sessions. Cette option n'a aucune influence si le contrôle du cache a été mis à `nocache`.

```
session.use_trans_sid = 1
```

Permet d'indiquer si l'identifiant de session doit être transmis de manière transparente de page en page (URL rewriting). (Pour les versions PHP<=4.1.1, cette option est utilisable uniquement si PHP a été compilé avec l'option *--enable-trans-sid*). Par défaut, cette option est activée.

```
session.hash_function = 0
```

Définit le format de hash. `0` pour MD5 (128 bits) ou `1` pour SHA-1 (160 bits)..

```
url_rewriter.tags = "a=href,area=href,frame=src,input=src,form=fakeentry"
```

Permet de définir quelles URL seront réécrites afin de transmettre l'identifiant de session, si l'option *session.use_trans_sid* est activée. Par défaut, ce sont les tags `a=href`, `area=href`, `frame=src`, `input=src`, `form=fakeentry`.

Génération du document

Cache et compression

```
output_buffering = Off
```

La mise en tampon (ou mémoire cache) vous permet de n'envoyer le document généré qu'une fois le script terminé. Cela permet notamment d'envoyer des en-têtes (instructions de manipulation des sessions et des cookies y compris), même après avoir commencé à générer la page. Mais, en contrepartie, cela ralentit la couche de sortie de PHP. Si vous voulez limiter le tampon à une certaine taille, vous pouvez spécifier une taille en octets à la place du `On` (ex. : `output_buffering=4096`).

```
output_handler =
```

Cette option permet d'appliquer une fonction (typiquement de compression) sur le document généré avant de l'envoyer au client. Par exemple, si vous configurez `output_handler` vers `ob_gzhanler`, la sortie sera compressée pour les navigateurs qui supportent Gzip. Si vous entrez un nom de fonction pour `output_handler`, cela activera automatiquement la mise en cache des sorties.

```
zlib.output_compression = Off
```

Cette directive permet la compression des sorties à l'aide de la bibliothèque `Zlib`. Les valeurs que vous pouvez indiquer pour cette commande peuvent être `On`, `Off` ou une taille spécifique de tampon à utiliser pour la compression (la taille par défaut est 4 Kb). Attention : `output_handler` doit être vide si cette directive est activée.

```
implicit_flush = Off
```

Implicit flush demande à PHP de vider les tampons de sortie dès qu'un élément du document est généré. Ceci est équivalent à l'appel de la fonction `flush()` après chaque commande `echo()` ou `print()` et après chaque bloc HTML. Si vous activez cette fonction, vous réduirez vos performances. Cette option est conseillée dans des objectifs de débogage.

Éléments par défaut

```
auto_prepend_file =
```

Pour ajouter automatiquement un fichier avant un document PHP.

```
auto_append_file =
```

Pour ajouter automatiquement un fichier après un document PHP.

```
default_mimetype = "text/html"
default_charset = "iso-8859-1"
```

Spécifie le type du document et l'encodage. Sachant que depuis PHP 4.0b4, PHP envoie par défaut un en-tête précisant l'encodage. Si vous ne voulez pas le préciser, vous devrez, ici, mettre une chaîne vide. Par défaut, s'il n'est pas précisé ici, le type du document est `text/html`.

Gestion de l'affichage

Nombres décimaux

```
precision    =  12
```

Indiquez le nombre de décimales après la virgule que vous souhaitez afficher.

Source

Couleurs pour la coloration syntaxique de votre code. Indiquez les codes couleurs qui vous conviennent :

```
highlight.string  = #CC0000
```

pour une chaîne de caractères ;

```
highlight.comment = #FF9900
```

pour un commentaire ;

```
highlight.keyword = #006600
```

pour un mot-clé ;

```
highlight.bg      = #FFFFFF
```

pour le fond ;

```
highlight.default = #0000CC
```

par défaut ;

```
highlight.html    = #000000
```

pour le html.

Extensions dynamiques

```
extension_dir = ./
```

Indiquez le chemin du répertoire dans lequel se trouvent les extensions (modules) chargeables.

Si vous souhaitez qu'une extension soit chargée automatiquement, utilisez la syntaxe suivante :

```
extension=modulename.extension
```

Par exemple, pour Windows :

```
extension=php_gd.dll
```

ou, sous UNIX :

```
extension=php_gd.so
```

Notez que vous ne devez indiquer que le nom du module, et non pas les informations concernant le chemin du répertoire (ce dernier doit obligatoirement être celui spécifié par `extension_dir`).

Configuration des extensions

BcMath

```
bcmath.scale = 0
```

Précision avec laquelle BcMath doit travailler (nombre de chiffres après la virgule souhaités).

Bases de données

```
sql.safe_mode = Off
```

Active ou non le mode sécurisé pour les bases de données.

Il existe de nombreux autres paramètres pour les bases de données. Ceux-ci seront évoqués dans les chapitres correspondants.

Sockets

```
sockets.use_system_read = On
```

Utilise la fonction système `read()` au lieu de la fonction programme `php_read()`.

Divers

Browscap

```
browscap = extra/browscap.ini
```

Chemin vers le fichier *browscap.ini* contenant les informations qui permettent de déduire les caractéristiques des navigateurs d'après leur `HTTP_USER_AGENT`.

2.3. Les éditeurs et débogueurs PHP

Dans le monde de la programmation, les éditeurs de texte font office de religions. On en utilise un comme on choisit une église. Comme les religions, les éditeurs connaissent leurs guerres. L'une des plus célèbres oppose les partisans de Vi aux zélateurs d'Emacs. Tout a été dit : "Emacs rend beau", "Vi fait repousser les cheveux", etc. En cherchant bien, on peut même trouver sur le Web un mémo sur les différents éditeurs de texte, rédigé pour une université canadienne, commençant par cette citation :

"Papa, pourquoi nous cachons-nous de la police ?

- Ils utilisent Emacs, fils, et nous utilisons Vi."

Les adeptes d'un éditeur en particulier ressentent, généralement, le besoin de prouver que le leur est le meilleur, le plus beau, le plus complet. Cela entraîne immanquablement de folles discussions fort peu constructives, mais folkloriques, voire divertissantes. Mais le débutant en quête de conseils ne s'y retrouve pas pour autant.

On peut tout de même regrouper les éditeurs en différentes grandes familles, pour ensuite comparer leurs avantages et inconvénients respectifs. Il n'est pas opportun de comparer Emacs à Dreamweaver MX. En revanche, mettre face à face PHPEdit et Komodo présente un intérêt. Sur le CD-ROM, vous trouverez un tableau comparatif des différents éditeurs ainsi que quelques fiches descriptives.

Les goûts et attentes de chacun interviennent beaucoup dans le choix d'un éditeur. Le contexte de développement de PHP joue également. Si c'est pour un projet isolé, le programmeur pourra choisir librement ; si, en revanche, il doit travailler en étroite collaboration avec un graphiste, un logiciel comme Dreamweaver MX sera plus approprié. Pour mettre sur pied une architecture complexe imbriquant de multiples pages en PHP, une solution comme Zend Studio ou Maguma rendra de bien meilleurs services.

Les outils les plus puissants ne doivent pas faire oublier qu'un site peut être développé avec un simple bloc-note (bien moins gourmand en mémoire et CPU). Ce n'est pas l'éditeur qui fait le développeur. Les assistants les plus poussés et conviviaux ne remplaceront pas une bonne connaissance du langage.

Nous avons classé les éditeurs en différentes catégories. Nous savons que ces catégories seront à coup sûr critiquées par les partisans de l'un ou l'autre des logiciels présentés, mais c'est un mal nécessaire…

- L'artillerie lourde, ce sont ces éditeurs qui intègrent PHP à toute une solution de développement web. Résolument tournés vers PHP, il leur faut une plateforme assez musclée pour fonctionner, et ils n'ont pas été pensés pour le développeur solitaire, coincé entre pizza et CVS. Ils permettent souvent un travail collaboratif poussé, qui place l'éditeur au centre d'une architecture incluant un serveur web et un serveur de bases de données. Leur prix souvent élevé les met hors de la portée des programmeurs isolés.
- Les spécialistes, ce sont des éditeurs PHP purement PHP. Ils ont été développés pour ce langage de script en particulier. Étant donné qu'ils sont assez nombreux, nous ne présentons ici que ceux que nous avons jugés pertinents en fonction de critères jugés primordiaux par plusieurs développeurs.
- Nous traitons enfin, dans une partie séparée, Dreamweaver et GoLive, deux logiciels de création de sites web qui gèrent désormais PHP, mais qui restent tout de même en deçà de ce qu'offrent les éditeurs spécialisés.

Nos plus vifs remerciements vont à Stéphane Pineau, auteur d'un comparatif d'éditeurs PHP, qui nous a permis de reproduire ici en grande partie son travail (par ailleurs disponible à l'adresse **http://steph.pineau.free.fr/php/index.php?LNK=EDIT/**). Nous vous conseillons d'ailleurs d'aller consulter cette page régulièrement pour vous tenir à jour des dernières mises à jour d'éditeurs. Vous pourrez également y effectuer des recherches multicritères pour choisir l'éditeur qui vous convient le plus.

L'artillerie lourde

Zend Studio

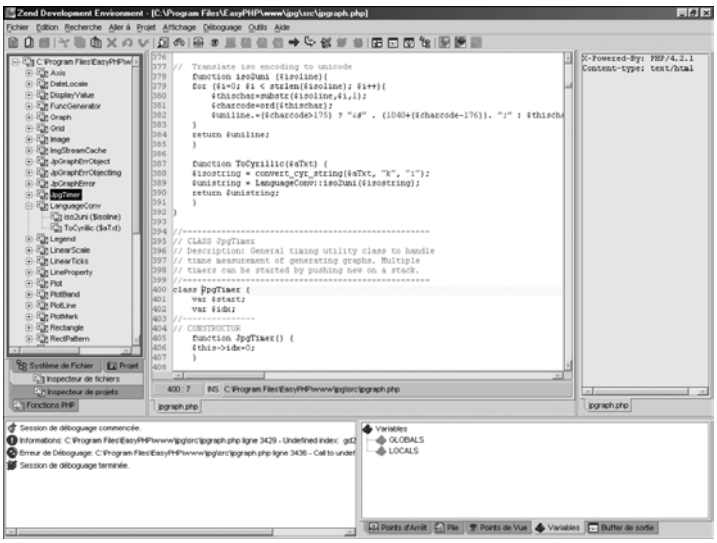


Figure 2.23 : *Zend Studio*

Tableau 2.1 : Zend Studio

Zend Studio	version 3.5.1
Éditeur	Zend Technologies Ltd.
Langues	Anglais, allemand, français, hollandais, espagnol, italien, coréen.
Plateformes	Windows, Linux, MacOSX.
Langages supportés	PHP, HTML.
Prix	249 $US.
URL	www.zend.com

Principales fonctionnalités et commentaire

L'éditeur offre toutes les fonctions de base nécessaires : complétion du code PHP et HTML, complétion également pour les variables, liste des fonctions disponibles dans un fichier, visualisation des fichiers en includes, coloration syntaxique, édition du même fichier depuis différentes fenêtres, édition de fichiers depuis un serveur FTP, etc. Le débogage est possible en local, et également en direct sur des serveurs distants. Cela permet ainsi de contrôler l'exécution de PHP sur les serveurs de production. Le débogueur intégré est particulièrement performant. La suite Zend inclut également un centre d'administration pour configurer et administrer les serveurs, ainsi qu'une application d'aide complète et fonctionnelle. On remarque aussi la présence du Zend Optimizer, logiciel optimisant le code PHP et permettant l'exécution de code camouflé (obfusqué). Il supporte la version 5 de PHP. Au rang des dernières nouveautés, on trouve les templates de code, personnalisation des schémas de coloration, débogage actif ou passif, transmission sécurisée des fichiers (publication par SFTP et FTP sur SSH). Certainement l'une des solutions les plus abouties et les plus complètes pour la création d'applications en PHP.

Maguma Studio

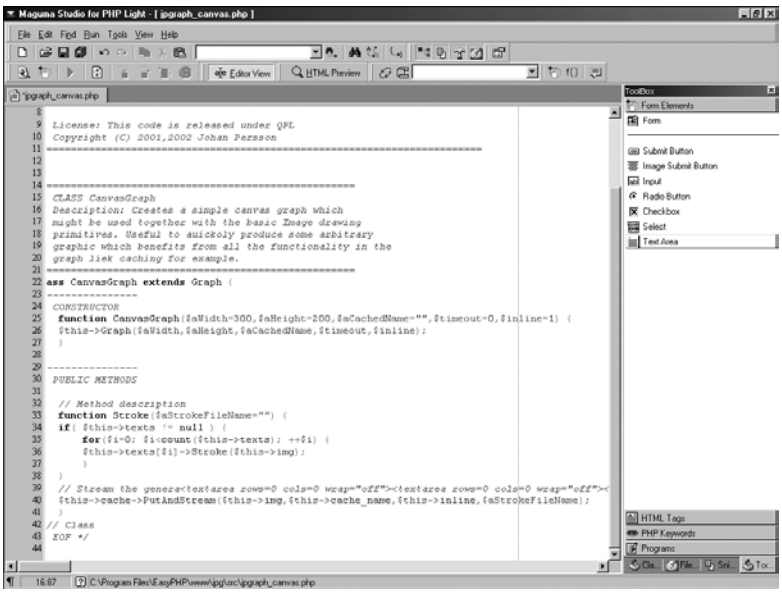


Figure 2.24 : *Maguma Studio*

Tableau 2.2 : Maguma Studio

Maguma Studio	version 1.3.2
Éditeur	Maguma.
Langue	Anglais.
OS	Windows.
Langage supporté	PHP.
Prix	69 € pour l'éditeur. Existe une version gratuite limitée.
URL	www.maguma.com

Principales fonctionnalités et commentaire
La plateforme Maguma intègre, en plus d'un éditeur puissant, toute une architecture de travail en commun, allant du client au serveur en passant par une interface d'administration générale.
L'éditeur offre la numérotation des lignes, la coloration syntaxique, une édition multi-fichiers, une aide PHP intégrée, une aide MySQL intégrée, une aide HTML intégrée, un gestionnaire de fichiers, un gestionnaire FTP, un visualiseur HTML interne, une visualisation/exécution depuis l'éditeur, un débogueur interne ou externe, un explorateur de code PHP, l'insertion assistée de code HTML, l'insertion assistée de code CSS, l'insertion assistée de code PHP, l'auto-complétion PHP, la gestion des abréviations (code template), le commentaire des fonctions/classes, la recherche des paires []{}() et le RegExp en mode recherche.

NuSphere PHPEd

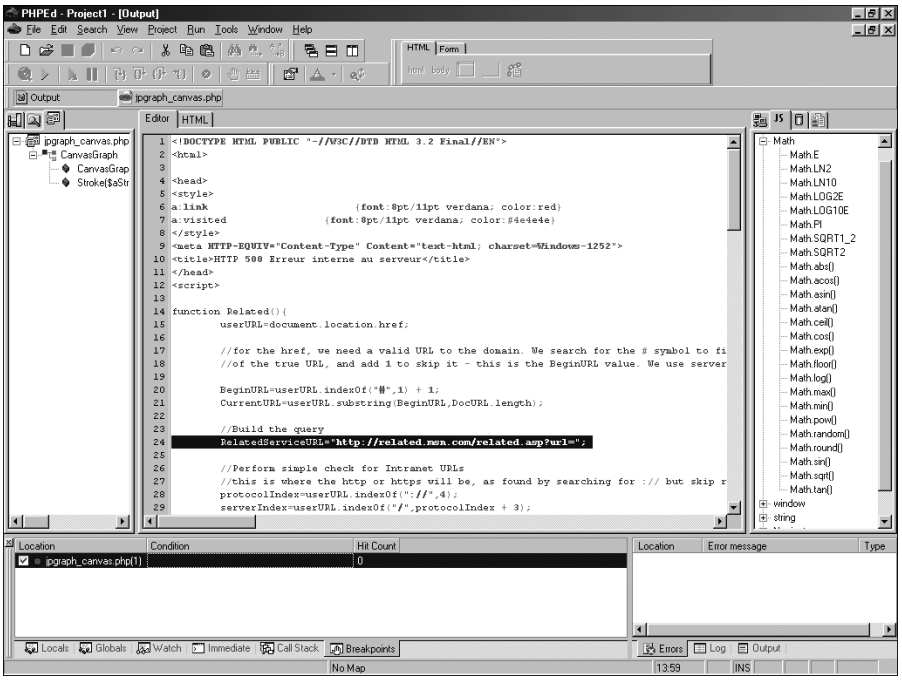


Figure 2.25 : *NuSphere PHPEd*

Tableau 2.3 : NuSphere PHPEd

NuSphere PHPEd	version 3.3
Éditeur	NuSphere.
Langue	Anglais.
Plateformes	Windows, Linux, Solaris
Langages supportés	PHP, HTML.
Prix	299 $US.
URL	www.nusphere.com/products/phpadv.htm
Principales fonctionnalités et commentaire Ajout de code depuis des modèles, débogueur de code, interfaçage avec des serveurs de bases de données (support en natif de PostgreSQL), gestion de projets, connexion en FTP, FTPS, WebDAV/HTTPS, gestion du travail en équipe, débogage de sessions, optimisation du code, etc. PhpED offre une complétion automatique du code qui supporte la programmation orientée objet. Le "PHP Profiler" intégré vous aide dans votre développement et permet de constater sa progression et l'optimisation de son code. Des fonctionnalités originales pour une solution vraiment complète.	

Les spécialistes

Tous les éditeurs présentés ici ont un débogueur intégré, à l'exception de Jext qui compense ce manque par une kyrielle de fonctionnalités et d'add-on.

Komodo

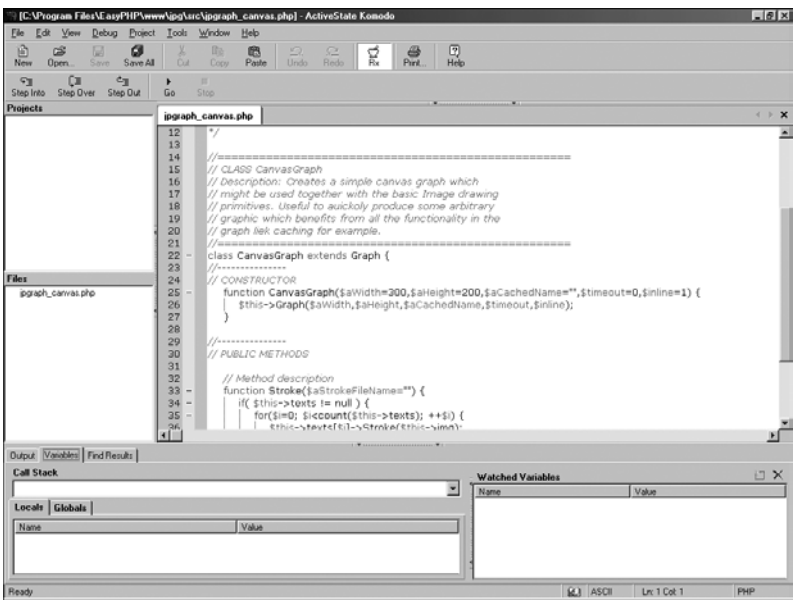


Figure 2.26 : *Komodo*

Tableau 2.4 : Komodo

Komodo	version 3
Éditeur	ActiveState.
Langue	Anglais.
Plateformes	Windows et Linux.
Langages supportés	C, C++, Diff, Eiffel, HTML, Java, Javascript, Latex, Lisp, PHP, Pascal, Perl, Python, tcl, vb, XML, XLT, etc.
Prix	La version 1.1 est gratuite dans le cadre d'une utilisation en milieu enseignant, ou à visée éducative. La version 1.2 est téléchargeable en évaluation pour 21 jours, puis payante. De 29,5 $US à 296 $US selon les types de licences.
URL	www.activestate.com/Products/Komodo/
Principales fonctionnalités et commentaire Numérotation des lignes, coloration syntaxique, multi-fichiers, aide PHP intégrée, aide MySQL intégrée, aide HTML intégrée, gestionnaire de projets, gestionnaire de fichiers, gestionnaire de favoris, gestionnaire FTP, liste de "reste à faire" (todo), rechargement automatique des derniers fichiers, visualiseur HTML interne, visualisation/exécution depuis l'éditeur, débogueur interne ou externe, macros, explorateur de code PHP, explorateur de code HTML, insertion assistée de code HTML, insertion assistée de code CSS, insertion assistée de code JavaScript, insertion assistée de code PHP, insertion code template, auto-complétion PHP, gestion des abréviations (code template), commentaire des fonctions/classes, recherche des paires []{}(), indentation / désindentation de blocs, commentaire de blocs, partage de la fenêtre d'édition, RegExp en mode recherche, archivage des fichiers/projets, sélecteur de couleurs. Développé autour du navigateur Mozilla auquel il s'intègre et de l'éditeur Scintilla. Un petit module permet de tester ses expressions régulières avec visualisation du résultat en temps réel. Possibilité de cacher le code des classes, fonctions et blocs (outlining). "Debug" pour Perl, Python, PHP et XSLT. Affichage des paramètres requis lors de l'utilisation d'une fonction. Rechargement des derniers fichiers, gestionnaire FTP. La version comporte son lot d'améliorations et d'ajouts de fonctionnalités (dont le support de PHP5). La liste des nouveautés est consultable à cette adresse : http://aspn.activestate.com/ASPN/docs/Komodo/3.0/relnotes.html#New_Features.	

PhpCoder

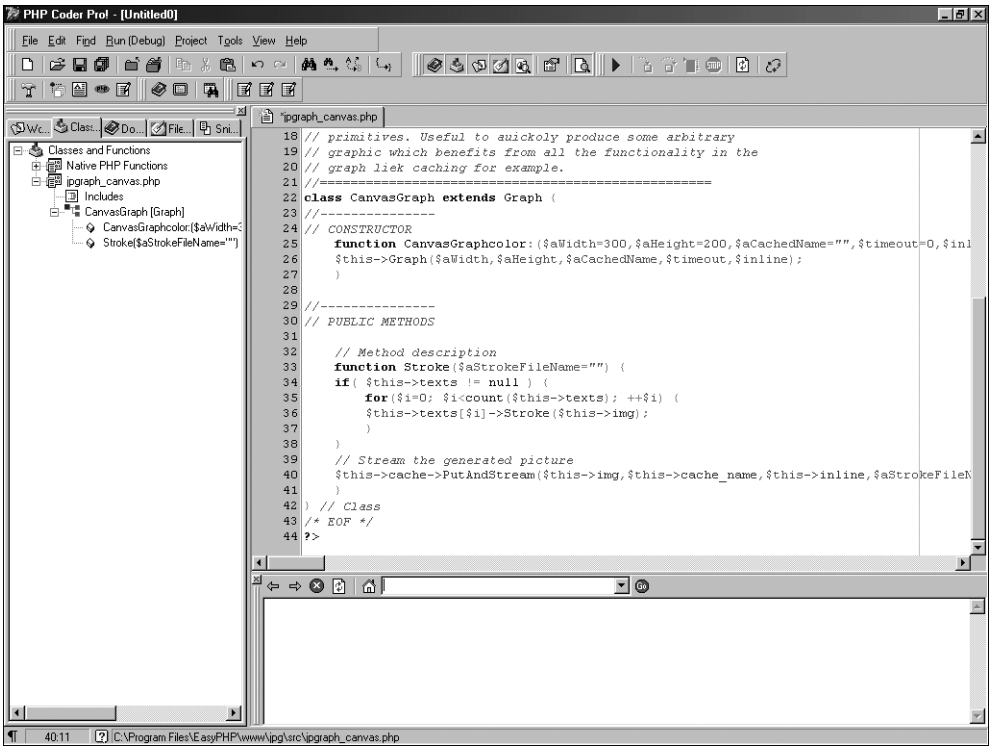


Figure 2.27 : *PhpCoder*

Tableau 2.5 : PhpCoder

PhpCoder	version R2 Prerealese 3
Éditeur	phpIDE.de.
Langue	Anglais.
Plateforme	Windows.
Langage supporté	PHP.
Prix	Gratuit.
URL	www.phpide.de
Principales fonctionnalités et commentaire Numérotation des lignes, coloration syntaxique, multi-fichiers, aide PHP intégrée, aide MySQL intégrée, gestionnaire de projets, visualiseur HTML interne, visualisation/exécution depuis l'éditeur, débogueur interne ou externe, insertion assistée de code HTML, l'auto-complétion PHP. Un clone de PHPEd quasi conforme, mais qui nécessite encore quelques améliorations.	

PHPEdit

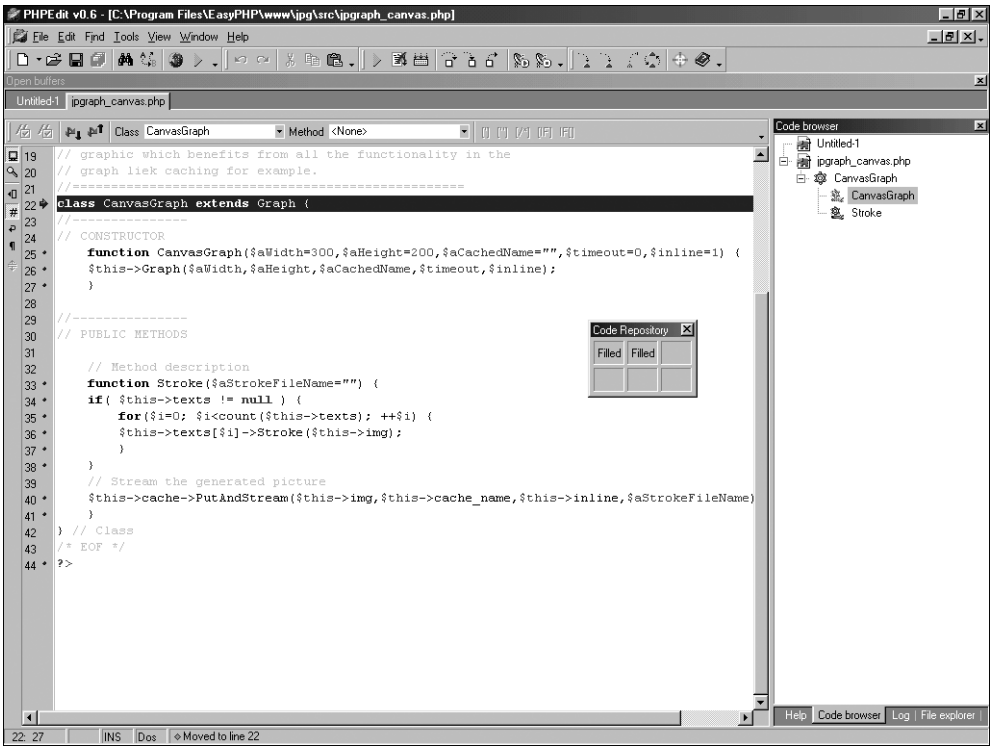


Figure 2.28 : *PHPEdit*

Tableau 2.6 : PHPEdit

PHPEdit	Version 1.0.3.68
Éditeurs	Waterproof SARL
Langue	Anglais, français, allemand, espagnol.
Plateforme	Windows et Linux.
Langage supporté	PHP.
Prix	Gratuit pour une utilisation personnelle, version professionnelle à 75 € HT (Open Source).
URL	**www.phpedit.com**

Principales fonctionnalités et commentaire
Resté longtemps à la version 0.6, PHPEdit a tenu ses promesses et comblé ses lacunes de jeunesse. Il est disponible en français avec une documentation complète et de très nombreuses fonctionnalités. Les dernières versions ont apporté un générateur d'aide, des raccourcis personnalisables, plus de 100 commandes scriptables, le mappage de clavier personnel, un gestionnaire de todo. L'éditeur offre également : numérotation des lignes, coloration syntaxique, multi-fichiers, aide PHP intégrée, aide MySQL intégrée, aide HTML intégrée, gestionnaire de fichiers, visualisation/exécution depuis l'éditeur,

PHPEdit	Version 1.0.3.68

débogueur interne ou externe, explorateur de code PHP, insertion assistée de code PHP, gestion des abréviations (code template), commentaire des fonctions/classes, indentation/désindentation de blocs, commentaire de blocs, RegExp en mode recherche L'explorateur de code est excellent ; il sépare bien les fonctions et classes de chaque fichier, et hiérarchise correctement les méthodes des classes. Appréciable également : la fonction d'*autocomment* des classes et fonctions. Le système d'insertion assistée de code PHP, similaire à celui du VBA, est agréable (plus que celui de PHPEd), et l'affichage des paramètres requis d'une fonction native par simple placement du curseur entre parenthèses est vraiment pratique. On apprécie également la possibilité d'imprimer avec les numéros de lignes et la mise en valeur des mots-clés. Un petit gestionnaire de copier/coller fonctionnant en *drag and drop* est intéressant. S'il était possible de l'intégrer en tant que panneau latéral plutôt que de le laisser en fenêtre libre, ce serait encore plus pratique. Dommage qu'aucun choix de langue ne soit proposé, d'autant qu'il s'agit d'une production française. L'absence de toute documentation est vraiment pénible, car nombre de petits gadgets assez déroutants de prime abord mériteraient une aide un peu plus substantielle.

SciTE

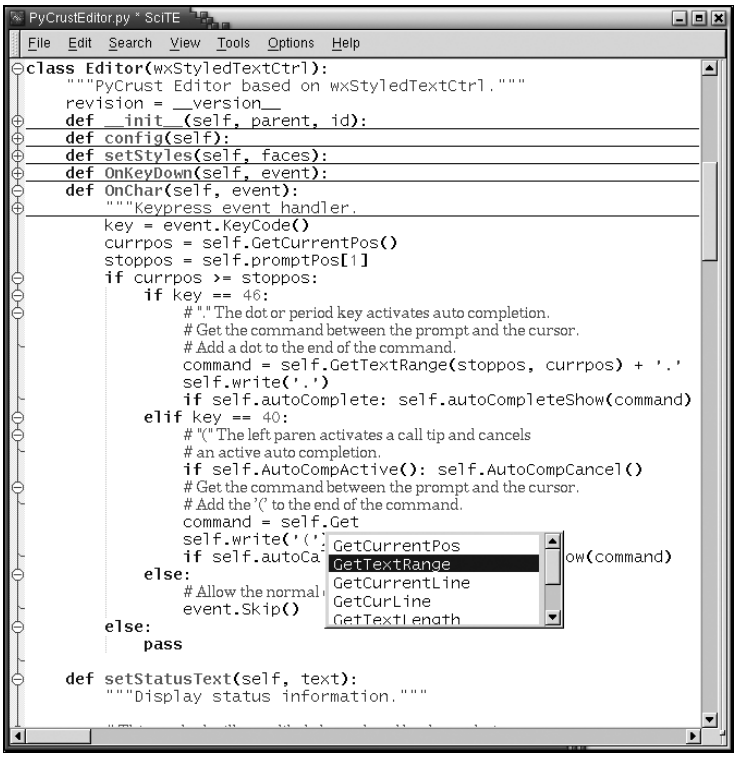


Figure 2.29 : *SciTE*

Tableau 2.7 : SciTe

SciTE	version 1.61
Éditeur	Neil Hodgson.
Langues	Anglais, Français, Allemand.
Plateformes	Windows et Linux.
Langages supportés	C, C++, C#, Eiffel, Java, Js, Latex, Lisp, Pascal, Perl, Php, Python, Sql, Vbscript, XML, etc.
Prix	Gratuit (Open Source).
URL	www.scintilla.org

Principales fonctionnalités et commentaire

Numérotation des lignes, coloration syntaxique, multi-fichiers, gestionnaire de projets, rechargement automatique des derniers fichiers, visualisation/exécution depuis l'éditeur, débogueur interne ou externe, macros, insertion assistée de code PHP, auto-complétion PHP, gestion des abréviations (code template), recherche des paires []{}(), commentaire de blocs, RegExp en mode recherche. L'éditeur offre également la recherche dans des fichiers non ouverts. Peut se placer en mode réduit dans la barre des tâches. Il ne pèse que 500 Ko avec sa DLL. On peut zoomer et dézoomer du code en cours d'édition. Permet de masquer l'affichage (outlining) logique des blocs d'un texte (HTML) ou d'un script. On peut donc masquer le corps des fonctions (d'un bloc `If/end`, par exemple) d'un simple clic (ou par Ctrl+*). On peut ainsi obtenir le squelette d'un script en n'affichant plus que le nom des classes, les fonctions et les commentaires. Il est également possible d'effectuer une inversion rapide des lignes (ligne courante avant la suivante). L'éditeur se configure via des fichiers, ce qui est un peu lourd. La configuration par défaut est plutôt restreinte ; nombre de possibilités ne sont pas visibles de prime abord. Les fichiers pour le français ne sont pas inclus dans l'archive, mais on peut les récupérer sur : www.scintilla.org/locale.fr.properties. Le chargement en mémoire est quasi instantané sur un PIII 533 MHz, ce qui est loin d'être le cas de bien des éditeurs.

On ne trouve pas de macros ni de gestion de projets en natif, mais ce manque est compensé par un add-on FilerX. Il n'existe pas d'explorateur de code. Trop de fonctions intéressantes sont désactivées par défaut. Lisez bien la documentation pour connaître toutes les possibilités de paramétrage. C'est vraiment l'éditeur à ne pas manquer pour sa légèreté et sa puissance, d'autant que les sources fournies (projet Scintilla) permettent de greffer ses propres fonctions. C'est d'ailleurs l'éditeur qui est intégré dans Komodo.

Voici quelques "trucs" pour activer des fonctions intéressantes. Pour activer l'outlining, il faut décommenter la ligne `fold.html=1` dans le fichier *html.properties*, et la ligne `fold=1` dans le fichier *sciteglobal.properties*. Dans ce dernier, vous pouvez également mettre la valeur 0 à la ligne `fold.symbol`. Pour afficher les numéros de lignes, il faut décommenter la ligne `line.number` dans le fichier *sciteglobal.properties*, et lui mettre une valeur de 30. Pour pouvoir ouvrir plusieurs fichiers, il faut décommenter la ligne `buffers` dans le fichier *sciteglobal.properties*, et lui mettre la valeur de 10 par exemple.

Jext

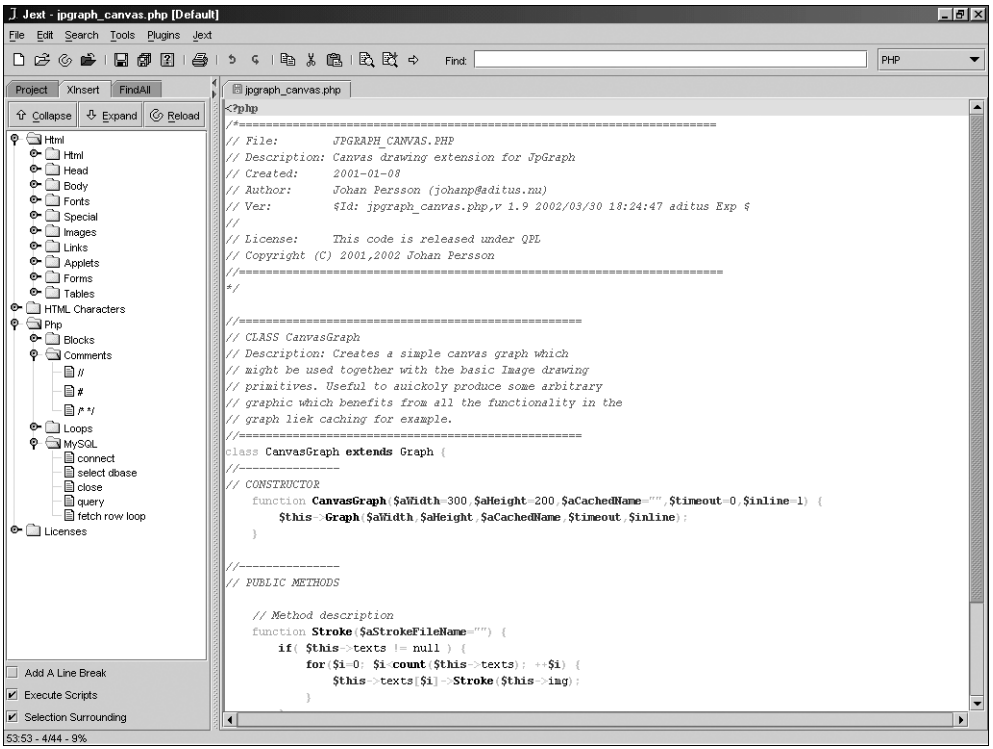


Figure 2.30 : *Jext*

Tableau 2.8 : Jext

Jext	version 5
Éditeur.	Romain Guy.
Langues.	Français, anglais.
Plateformes	Java (multi-plateforme). Nécessite JRE 1.301 ou JDK 1.2.
Langages supportés	PHP, ASM, ASP, C, C#, C++, Eiffel, HTML, XML, Java, Perl, Python, Sql, Shell scripts, etc.
Prix	Gratuit (OpenSource).
URL	www.jext.org

Principales fonctionnalités et commentaire
Numérotation des lignes, coloration syntaxique, multi-fichiers, gestionnaire de projets (add-on), gestionnaire de fichiers, gestionnaire de favoris, gestionnaire FTP (add-on), rechargement automatique des derniers fichiers, explorateur de code PHP, insertion assistée de code HTML, insertion assistée de code PHP, auto-complétion PHP, gestion des abréviations (code template), recherche des paires []{}(), indentation/désindentation de blocs, commentaire de blocs, RegExp en mode recherche.

Jext	version 5

Avec ses add-on, cet éditeur est particulièrement intéressant. Il offre ainsi un système de bureau, c'est-à-dire une liste de fichiers ouverts faisant partie, par exemple, d'un même programme. Il est possible de créer plusieurs bureaux, et donc de disposer d'un gestionnaire de projets basique mais efficace. Un add-on "Wapppaaa" permet d'ouvrir en une opération l'ensemble des fichiers contenus dans un répertoire et ses sous-répertoires. L'add-on Xinsert permet de gérer ses code template de manière hiérarchique, et cela pour chaque langage voulu. Il autorise l'inclusion automatique du texte sélectionné dans le template choisi. L'add-on FindAll permet, en double-cliquant sur un mot, et en appuyant sur Ctrl + F3, d'obtenir toutes ses occurences dans une liste sélectionnable ; c'est très utile pour trouver les occurrences d'une variable ou d'une fonction. Les recherches sont, de plus, mémorisées. L'add-on ProjectMaster remplacera utilement le gestionnaire de projets CodeMaster fourni par défaut, car il s'intègre directement dans le panneau latéral. Bien que plutôt axé Java, il peut être utilisé en PHP. Jext dispose d'un langage de script intégré, Dawn. On appréciera également le système One Clic : il suffit de sélectionner une commande de ce menu, par exemple **Ajouter un commentaire simple**, puis de cliquer sur chaque ligne que l'on veut commenter, et c'est tout. La commande sélectionnée s'applique à chaque ligne cliquée tant que l'option *One Click* n'est pas arrêtée. L'add-on Code2Html transforme le fichier en cours (ou la sélection) en fichier HTML, tout en conservant la coloration syntaxique. Idéal pour publier le code source d'un langage quelconque de manière attrayante. De nombreux autres add-on sont disponibles : console système, skins, impression avec numéros de lignes, tri de lignes... En bref : c'est un excellent éditeur. Il n'est pas inutile de lire la documentation (en français ou en anglais) pour tirer parti de toutes ses possibilités.


REMARQUE

Le choix des auteurs

À titre purement indicatif, les auteurs de cette Bible utilisent, par ordre de pagaille : Vi, Emacs, Nedit, Wordpad, le bloc-notes ou encore UltraEdit pour programmer en PHP. Vous le voyez, aucun éditeur spécialisé... Comme quoi, ce n'est (décidément !) vraiment pas l'éditeur qui fait le développeur.

Dreamweaver et GoLive

Au départ, ce sont des outils de création de sites web destinés aux graphistes et webmestres Aujourd'hui, et tout particulièrement dans les dernières versions, les deux éditeurs supportent particulièrement bien PHP. On peut, par exemple, facilement créer des pages contenant des objets en PHP en puisant dans des bibliothèques prêtes à l'emploi. Les deux éditeurs ont intégré le travail avec des bases de données (GoLive est vendu avec Apache, PHP, JSP et MySQL). Sans aller jusqu'au serveur de débogage de Zend, à installer directement sur les machines de production, Dreamweaver et GoLive ont dépassé le simple stade du "wysiwyg" (what you see is what you get ou, vous voyez ce que vous avez). Les deux logiciels disposent d'une composante de travail collaboratif. Il est même possible de travailler avec Webdav, ou même d'avoir une gestion centralisée du code source.

L'intérêt des logiciels de création de sites web dans le cadre d'un développement en PHP reste limité. Si une page contient uniquement un développement original en PHP, il sera inutile d'aller l'éditer dans Dreamweaver. De même, quel intérêt peut-il y avoir à développer toute une application en PHP dans un logiciel comme GoLive, largement plus performant dans le Webdesign ?

Dans une optique PHP, Dreamweaver et GoLive restent de très bons outils d'intégration finale et d'habillage des développements en PHP. Un meilleur support de PHP reste un point positif. Cela signifie, par exemple, qu'un graphiste aura plus de souplesse pour travailler une page contenant du code PHP.

Quelques site d'éditeurs
Document traitant des éditeurs, rédigé pour l'université McGill au Canada :
www.cs.mcgill.ca/~navindra/editors/
Vi :
http://vim.sourceforge.net/
Emacs :
www.gnu.org/software/emacs/emacs.html
UltraEdit :
www.ultraedit.com/

Pages personnelles de Stéphane Pineau
Dictionnaire francophone des acronymes informatiques :
www.teaser.fr/~spineau/acrodict/index.htm
Script PHP 3 gratuits (forum, gestionnaires BDD, etc.) :
http://steph.pineau.free.fr/php/index.php3

Chapitre 3

Le langage PHP

3.1. Intégrer le code PHP au HTML

Comme cela a déjà été dit dans l'introduction, le langage PHP est un langage de script qui s'insère dans des pages HTML, et les parties écrites en langage PHP sont déclarées au moyen de balises reconnues par le serveur.

Le code PHP s'écrit donc avec un éditeur de texte et l'utilisation des éditeurs "wysiwyg" (what you see is what you get) s'arrête lorsque l'écriture du langage PHP commence.

Vous pouvez vous reporter à la section "Les éditeurs" pour choisir le vôtre.

Le principe de fonctionnement de PHP est assez simple : le serveur va effectuer un travail d'interprétation avant de retourner le résultat (généralement une page HTML) au client (classiquement un navigateur Internet).

L'avantage par rapport à l'HTML est alors considérable, puisqu'il est ainsi possible de concevoir des sites dits dynamiques. C'est-à-dire que, d'une fois sur l'autre, une page peut changer de contenu sans qu'aucune intervention humaine ne soit nécessaire.

Prenons le cas d'un forum : il est souhaitable que, dès qu'une personne soumet une question, celle-ci soit disponible sur le site. C'est un cas typique d'utilisation d'un site Internet dynamique. Le langage PHP permettra alors de créer une page HTML à partir de ce message (et des autres messages disponibles dans la base de données).

Il faut toujours garder à l'esprit que c'est un langage de script qui est exécuté côté du serveur ; il ne dépend donc ni du navigateur ni du système d'exploitation du visiteur. L'avantage est indéniable puisque ce langage permet, à partir d'informations de nature différente, (base de données, heure actuelle, configuration du client, etc.) d'envoyer des pages dont on peut être sûr qu'elles seront compréhensibles par le navigateur du client (à condition de programmer proprement).

Les balises

Les parties correspondant au code PHP sont déclarées au moyen de balises. Il existe plusieurs types de balises en fonction de vos préférences, habitudes, et de la configuration de PHP sur le serveur ou tournera vos scripts.

Il existe quatre différents types de balises regroupés dans le tableau ci-après :

Tableau 3.1 : Les différentes balises

Balise	Remarque
`<?php … ?>`	Ce type de balise est le plus courant. Il est accepté par défaut par l'interpréteur.
`<script language="php">… </script>`	Ce type de balise est également accepté par défaut par l'interpréteur.

Balise	Remarque
`<? … ?>`	Ces balises courtes sont appelées “short tags”. Elles ne sont pas acceptées par défaut, cela fait partie de la configuration de PHP.
`<% … %>`	Ces balises sont là pour obtenir les mêmes balises que pour l’ASP. Elles ne sont pas acceptées par défaut, cela fait partie de la configuration de PHP.


RENVOI

Vous pouvez vous reporter au chapitre "Prise en main" afin de voir comment configurer PHP pour utiliser les balises qui vous serviront.

Pour ceux qui veulent en écrire le moins possible, il existe aussi deux autres types de balises qui ont pour effet de simplifier la syntaxe lorsque vous souhaitez simplement afficher du texte avec la commande `echo`.

Ainsi `<? echo "toto" ?>` peut être avantageusement remplacé par `<?= "toto" ?>` et `<% echo "toto" %>` par `<%= "toto" %>`.


CONSEIL

Ne vous éloignez pas des balises... standard
Si vous écrivez vos premiers scripts, utilisez le premier type de balises (`<?php… ?>`). En effet, celui-ci est compris de l’ensemble des serveurs, car il ne dépend pas de leur configuration. Le troisième est le type pour fainéants, mais peut se retourner contre vous… car, si vous ne pouvez pas accéder à la configuration du serveur après en avoir changé, il vous faudra modifier toutes les balises ! De même, il ne faut pas oublier que, si vous voulez distribuer votre script, il vous faut un script qui fonctionne sur n’importe quel serveur muni de PHP.

Mon premier script

Le premier script PHP est un grand classique ; il ne fait qu’afficher une phrase.

```
<html>
  <head>
    <title>Mon premier script</title>
  </head>
  <body>
  <?php
   echo "Mon premier script qui n'affiche
         que du texte en attendant mieux...";
 ?>
  </body>
</html>
```

Ici, à l'intérieur d'un script HTML classique, on a inséré des balises PHP dans lesquelles on a placé une seule instruction, `echo`, qui affiche simplement la chaîne de caractères qui suit.

Le séparateur d'instructions est classiquement le point-virgule ; il peut être omis après la dernière instruction d'un bloc PHP (ici, en l'occurrence, il aurait pu être omis).

Ce fichier, qui sera stocké sur le serveur, va être interprété puis envoyé au client (le navigateur Internet du visiteur) quand celui-ci y fera appel.

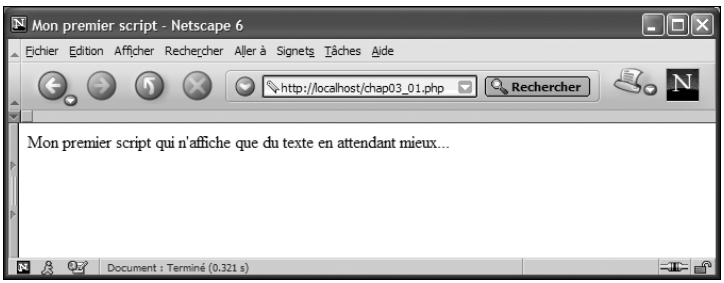


Figure 3.1 : *Ce que verra le visiteur*

Comme nous pouvons le constater en réclamant le code source du document reçu, le résultat de l'interprétation ne laisse plus apparaître de code PHP.

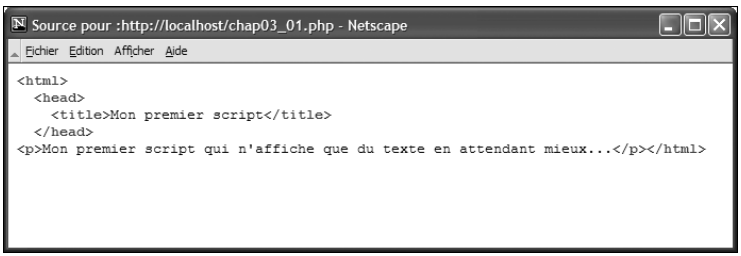


Figure 3.2 : *Le code source reçu par le client*

Les balises PHP peuvent être placées n'importe où dans le script ; le script suivant donnera le même résultat :

```
<html>
  <head>
    <title>Mon premier script</title>
  </head>
  <body>
  <?php
  echo "Mon premier script"
 ?> qui n'affiche <?php
  echo "que du texte en attendant mieux...";
 ?>
  </body>
</html>
```

Dans le cas présent, l'utilité n'est pas flagrante, mais nous verrons des cas d'applications lorsque nous étudierons les boucles de contrôle.


REMARQUE

Client-serveur

Il faut bien comprendre la logique client/serveur : d'un côté le serveur est chargé d'interpréter le code PHP et d'envoyer au client (le navigateur Internet du visiteur) le résultat de l'interprétation. De l'autre, le client se contente d'afficher la page correspondant au code source envoyé.

Il n'est donc pas question de demander à PHP d'agir sur le client. Vous ne trouverez, par exemple, aucune fonction permettant de suivre le déplacement de la souris, et toute modification du contenu de la page passe nécessairement par une nouvelle interrogation du serveur (i.e. affichage d'une nouvelle page ou, éventuellement, réaffichage de la page avec de nouveaux paramètres).

En contrepartie, le client n'a pas à interpréter PHP, ce qui implique que la machine n'a pas besoin d'outils spécifiques, ni de beaucoup de ressources : c'est le serveur qui fait tout le travail !

3.2. Les commentaires

Pour commenter ses scripts et permettre ainsi de les rendre plus clairs, il existe trois façons de faire :

Tableau 3.2 : Les différentes façons d'insérer des commentaires

Notation	Remarque
`# commentaire`	Le texte entre le signe `#` et la fin de la ligne sera en commentaire. Cette façon de noter est celle des scripts shell.
`// commentaire`	Le texte entre `//` et la fin de la ligne sera en commentaire.
`/* commentaire */`	Tout le texte entre `/*` et `*/` sera en commentaire.

Listing 3.1 : Exemple de script avec commentaires

```
<?php
echo "Ligne suivi d'un commentaire"; # commentaire façon shell
echo "Ligne suivi d'un commentaire";// commentaire façon C
/* commentaire
    sur
    plusieurs
    lignes
*/
echo "Ligne entre 2 blocs de commentaires";
/* commentaire
*    sur
*    plusieurs
```

```
*    lignes
*    plus lisibles
*/
/* cette
    partie
    est commentée
    // ce type de commentaire bien qu'inutile ici est valable
    # celui ci aussi peut être imbriqué
*/
?>
```

Un petit commentaire ?
Les commentaires sont importants pour s'y retrouver ; ils le sont plus encore quand vous relisez un script qui date de plusieurs mois. Mais cela n'a rien de nouveau et est valable pour n'importe quel langage…
Privilégiez l'utilisation des `//` et `/* */` et évitez `#`.

Ne pas imbriquer les commentaires /* */
Vous ne pouvez pas placer `*/` à l'intérieur d'un commentaire entre `/*` et `*/`. Par exemple, le script suivant provoquera une erreur :

```
<?php
/* je mets des remarques parce que je le veux bien
    /* j'aime tellement les remarques que je les imbrique */
*/
?>
```

Au moment de l'interprétation, les commentaires sont supprimés ; ils ne sont donc pas envoyés au client, et, ainsi, ne ralentissent en rien l'envoi de la page. Encore une bonne raison pour user et abuser des commentaires…

3.3. Les constantes

PHP permet de définir des constantes, autrement dit des chaînes de caractères représentant une valeur figée (chaîne de caractères, nombre ou booléen). Les constantes se déclarent par l'instruction `define()`.

Ainsi,

```
define("LANGAGE", "PHP");
define("VERSION", 4);
```

crée deux constantes, LANGAGE et VERSION, qui peuvent être utilisées très naturellement ;

```
echo LANGAGE." ".VERSION;
```

retournera ainsi

PHP 4

Outre les constantes que vous pourrez définir, PHP propose ses propres constantes dont :

- `PHP_VERSION` qui contient le numéro de version de PHP (ex. : "4.3.2").
- `PHP_OS` qui contient le nom du système d'exploitation sous lequel PHP tourne ("Linux", "WINNT", "WIN32", "HP-UX", "AIX", etc.).
- `__FILE__` qui contient le chemin complet du script actuellement lu (s'il s'agit d'un script inclus, c'est donc bien le chemin complet du script qui est retourné et non celui du script appelé).
- `__LINE__` qui contient le numéro de la ligne actuellement exécutée.

Existent aussi les constantes `TRUE` (vrai) et `FALSE` (faux) et les constantes de niveau d'erreur que nous présenterons plus loin.

Les constantes sont accessibles depuis n'importe quel endroit du code, y compris à l'intérieur des fonctions. Voir à ce sujet la section concernant "les fonctions" et la portée des variables.

3.4. Les variables

Définition et syntaxe

Les variables sont destinées à stocker les données qui seront utilisées et pourront être modifiées lors de l'exécution du programme. Le langage PHP, tout comme le langage Perl, utilise une forme particulière de syntaxe pour définir une variable. Les variables sont représentées avec un caractère '`$`' préfixant l'identification de cette variable. Par exemple, pour déclarer une nouvelle variable de nom "`maVariable`", il suffit de l'appeler `$maVariable` à l'intérieur du code source.

Contrairement au langage comme le C, PHP n'exige pas la déclaration préalable des identificateurs avant leur utilisation. Les variables sont donc dites non typées, c'est-à-dire qu'une variable peut prendre une valeur de chaîne de caractères et, ensuite, être utilisée pour contenir un entier. Le type est défini à l'affectation de cette variable. C'est en partie cette grande souplesse d'utilisation qui fait de PHP un langage simple et rapide d'accès.

Dans l'exemple ci-dessous, vous pouvez constater la simplicité d'utilisation des variables :

```
<?php
$maVariable = "0";
// $maVariable est une chaîne de caractères

$maVariable = $maVariable + 1;
echo $maVariable;
// Renvoie "1" et est maintenant un entier
```

```
$maVariable = 1;
// $maVariable est à present un entier

$maVariable = $maVariable / 2;
// $maVariable est maintenant du type double

$maVariable = 1 + "2 Vive le PHP !";
echo $maVariable;
// $maVariable est du type entier et renvoie 3
?>
```

Le nom de la variable peut être défini par tous les caractères alphanumériques ainsi que par le caractère `'_'` , mais il ne peut pas commencer par un chiffre.

`$maVariable`, `$_maVariable`, `$maVariable1` sont des variables correctes. En revanche, `$1maVariable` provoquera une erreur de syntaxe.

Faites chauffer la colle (attention à la casse…)
Les variables définies dans un programme PHP sont sensibles à la casse. Faites donc attention, dans vos programmes, à ne pas initialiser `$maVariable` et appeler `$MAVARIABLE` à l'utilisation. Ces deux variables sont distinctes.

Les variables dynamiques

Les variables dynamiques (aussi appelées variables "variables") sont des variables dont le nom dépend d'autres variables. Les noms de ces variables sont donc construits dynamiquement pendant l'exécution du code PHP.

L'exemple ci-dessous est bien plus parlant qu'un long discours :

```
<?php
$var = "Je développe en PHP.";
$dyn = "var";
echo $$dyn;  // Affiche "Je développe en PHP.";
?>
```

Ainsi, en écrivant simplement `$$dyn` on récupère la valeur de la variable dont le nom est contenu dans `$dyn`, c'est-à-dire `$var`.

Le recours aux variables dynamiques n'est généralement pas nécessaire, car elles peuvent, bien souvent, être avantageusement remplacées par l'utilisation de tableaux, mais qui sait... il est possible que cela vous dépanne un jour !

Il est souvent préférable, et parfois indispensable, de mettre le nom des variables entre accolades. Cela vous permettra, par exemple, de créer un nom de variable avec une partie variable et une partie fixe, comme c'est le cas dans l'exemple suivant :

```
<?php
$prefixe= "PHP";
$suffixe= " c'est sympa";
```

```
$dyn = "pre";
echo ${$dyn}fixe;     // Affichera PHP
$dyn = "suff";
echo ${$dyn}fixe;     // Affichera c'est sympa
?>
```

Le résultat aurait été évidemment tout autre si nous avions utilisé `$dynfixe`. Le problème est identique lorsqu'il s'agit de sélectionner un élément d'un tableau ; `$$dyn[0]` est ambigu, contrairement à `${$dyn[0]}` et `${$dyn}[0]`.

Les types

Comme nous avons pu le constater, les variables PHP n'ont pas un type prédéfini. Pourtant, un développeur peut être amené à manipuler les types pour le besoin d'une application et les problèmes de conversion peuvent non pas causer des erreurs, mais retourner des résultats surprenants et gênants. Pour éviter cela, le langage PHP possède un jeu de fonctions qui lui permet de fixer et de récupérer le type d'une variable donnée.

On peut classer les différents types possibles en trois catégories : les types scalaires, les types composés et les types spéciaux.

Tableau 3.3 : Définitions des quatre types scalaires

Type	Description	Exemple

`int` (ou `integer`)	Le type `int` est utilisé pour contenir des entiers (nombres sans virgule). Les valeurs peuvent aller de –2147483648 à 2147483647.	`$maVariable = 2;`
		`$maVariable = -4;`
		`$maVariable = 2002;`

`double` (ou `real` ou `float`)	Le type `double` sert à définir des nombres décimaux, c'est-à-dire comportant une virgule flottante.	`$maVariable = 1.0;`
		`$maVariable = -0.1`
		`$maVariable = 3.1415972;`

`string`	Le type `string` définit une chaîne de caractères.	`$maVariable = "Vive le php !";` `$maVariable = "1";`
`boolean` (ou `bool`)	Le type booléen définit une variable prenant des valeurs de type binaire : `TRUE` (Vrai) ou `FALSE` (Faux).	`$maVariable = TRUE;` `$maVariable = FALSE;`


REMARQUE

Notation dans les différentes bases

Le type entier peut prendre des valeurs exprimées en décimales (en base 10), hexadécimales (en base 16) et octales (en base 8). Pour indiquer le changement de base, le programmeur placera un 0 devant l'affectation.

```
<?php
$dec = 16;     // Affectation classique en base 10
```


```
$hex = 0x10;   // Le 0x indique l'affectation en hexadécimal
$oct = 020;    // Le simple 0 devant indique l'affectation en Octal
?>
```

`$dec`, `$hex` et `$oct` possèdent la même valeur mais exprimée dans une base différente.
Il est à noter qu'il n'est pas possible d'exprimer un nombre directement dans sa forme binaire.

Tableau 3.4 : Les deux types composés

Type	Description	Exemple
`array`	Désigne un type tableau (ensemble de valeurs). Voir la section *Tableaux*.	`$monTableau = array (2, "Super, du PHP !", "clé"=>"valeur");` `$monTableau[2] = "Top";`

`object`	Désigne un type objet (variable possédant ses propres propriétés, attributs et méthodes).
	Voir la section *Classes*.
		`class MaClasse`
		`{`
		`}`
		`$monObjet = new MaClasse()`

Tableau 3.5 : Les deux types spéciaux supportés

Type	Description
`resource`	Représente une référence (ex. : identifiant de connexion à une base de données ou toute autre source).
`NULL`	La variable de type `NULL` représente une variable ne possédant pas de valeur.

Rencontre avec le 9e type
Dans les descriptions des fonctions, nous utiliserons également, en lieu et place de certains types, d'autres termes qui ne correspondent pas véritablement à des types. Ainsi :

- *`mixed` désignera les paramètres (ou valeurs retour) pouvant être de différents types.*
- *`function` désignera des noms de fonctions (en fait, des types `string` mais avec une signification particulière).*
- *`numeric` désignera des paramètres (ou valeurs retour) de type `int` ou `double`.*
- *`void` désignera les fonctions sans paramètre ou sans valeur retour.*

Bien que le type des variables soit défini à leur initialisation, il est quand même possible de forcer une variable dans un type donné. Pour cela, nous utilisons la fonction `setType()`.

setType()

Affecte un type à une variable.

Syntaxe	`boolean setType(mixed $variable, string $type)`
`$variable`	Variable dont on veut fixer le type.
`$type`	Chaîne de caractères précisant le type que l'on veut affecter à la variable. Ce peut être au choix : `integer` pour un type entier. `double` pour un type réel. `string` pour une chaîne de caractères. `array` pour un tableau.
`retour`	`TRUE` en cas de succès ou `FALSE` (ex. : nom de type invalide).

Pour récupérer le type vous utiliserez la fonction `gettype()`.

getType()

Retourne le type de la variable.

Syntaxe	`string getType(mixed $variable)`
`$variable`	Variable dont on veut déterminer le type.
`retour`	Les différentes valeurs retournées possibles sont : `integer` pour un type entier. `double` pour un type réel. `string` pour une chaîne de caractères. `array` pour un tableau. `object` pour un objet. `resource` pour une ressource (un pointeur sur ...). `user function` pour une fonction créée par l'utilisateur. `unknown type` pour un type indéterminé.

Le code ci-dessous est un bon exemple de cette utilisation :

```
<?php
setType($toto, "integer");
// La variable est du type entier

echo getType ($toto);
// Renvoie "integer"
```

```
$i = 2+2;
echo getType ($i);
// Renvoie "integer"

$i = "J'aime le chiffre ".$i; // Concatenation d'une chaîne avec un entier
echo getType ($i);
// Renvoie "string"
?>
```

Le transtypage

Pour remédier aux problèmes que vous pouvez rencontrer dans vos manipulations de conversion, le développeur a la possibilité d'utiliser le transtypage (ou type `casting`). Sous ce terme un peu barbare, se cache la possibilité de créer une nouvelle variable contenant la valeur d'une autre avec un type différent mais compatible. Ainsi, le transtypage permet à un programmeur de modifier le type de ses variables pendant l'exécution de son programme, que ce soit pour faire un test d'égalité ou des opérations entre deux variables de type différent.

Le transtypage s'utilise, comme pour le langage C, en écrivant le type entre parenthèses devant le nom de la variable, comme indiqué dans l'exemple ci-dessous :

```
<?php
$monEntier = 10;
// $monEntier est un "integer"

$maChaine = (string)$monEntier;
// $maChaine est une chaine de caractères
?>
```

Les différentes conversions possibles sont donc :

- `(array)` pour convertir en type tableau.
- `(boolean)` ou `(bool)` pour convertir en type booléen.
- `(double)`, `(float)` ou `(real)` pour convertir en type réel (décimal).
- `(int)` ou `(integer)` pour convertir en type entier.
- `(object)` pour convertir en type objet.
- `(string)` pour convertir en type chaîne de caractères.

Dans l'exemple suivant, nous montrons comment, depuis la conversion en tableau ou en objet, un programmeur peut récupérer les données :

```
<?php
$maVariable = "Super, encore du PHP !";
$tableau = (array)$maVariable;
echo $tableau[0];
// renvoie "Super, encore du PHP !"

$objet = (object)$maVariable;
echo $objet->scalar;
```

```
// renvoie "Super, encore du PHP !"
?>
```

Notez que la conversion d'une variable de type scalaire (`int`, `double` ou `string`) dans le type objet créera un attribut nommé `scalar` et contenant la valeur de la variable.

Les références

PHP permet de créer une référence à une variable existante. De la même manière qu'avec le langage C, vous pouvez accéder directement à une zone mémoire contenant une variable. La référence à une variable permet de créer une nouvelle variable qui utilise la même zone mémoire que la variable d'origine.

Pour cela, vous devez l'indiquer à l'aide du caractère '`&`' devant le nom de la variable, comme le montre l'exemple suivant :

```
<?php
$var1 = "Bonjour";
$var2 = &$var1;
// $var1 et $var2 sont deux mêmes noms pour une même variable

$var2 = "Au revoir";
echo $var1;
// Affiche Au revoir
?>
```

Tester une variable

Dans vos manipulations des données, vous pouvez être amené à tester l'existence ou le type des variables que vous utilisez. Pour cela, on utilise un jeu de fonctions très utile.

isSet()

Détermine si une variable existe (a été initialisée).

Syntaxe	`boolean isSet(mixed $variable)`
`$variable`	Variable que l'on veut tester.
`retour`	`TRUE` si la variable existe bien et `FALSE` sinon.

unset()

Détruit les variables.

Syntaxe	`void unset(mixed $variable1 [, mixed $variable2 [, ...]])`
`$variable1, ...`	Les variables à détruire.

```
<?php
echo isSet($toto);    // Renvoie FALSE (ce qui n'affiche rien)
$toto = "";
echo isSet($toto);    // Renvoie TRUE (ce qui affiche 1)

unset ($toto);
echo isSet($toto);    // Renvoie FALSE (ce qui n'affiche rien)
?>
```

De la même manière, la fonction `empty()` détermine si une variable possède une valeur non nulle ou non.

empty()

Détermine si une variable possède une valeur non nulle ou non.

Syntaxe	`boolean empty(mixed $variable)`
`$variable`	Variable que l'on veut tester.
`retour`	`TRUE` si la variable n'existe pas, est une chaîne vide (`''`) ou vaut 0, `NULL`, `FALSE` sinon.

```
<?php
echo '<html><body>';

// affectation des variables
$var1 = 0; $var2 = 1; $var3 =""; $var4 ="Bonjour !";

// empty($var1) renvoie TRUE
if (empty($var1)) echo '$var1 est "vide"<br>';
else echo '$var1 = '.$var1.'<br>';

// empty($var2) renvoie FALSE
if (empty($var2)) echo '$var2 est "vide"<br>';
else echo '$var2 = '.$var2.'<br>';

// empty($var3) renvoie TRUE
if (empty($var3)) echo '$var3 est "vide"<br>';
else echo '$var3 = '.$var3.'<br>';

// empty($var4) renvoie FALSE
if (empty($var4)) echo '$var4 est "vide"<br>';
else echo '$var4 = '.$var4.'<br>';

echo '</body></html>';
?>
```

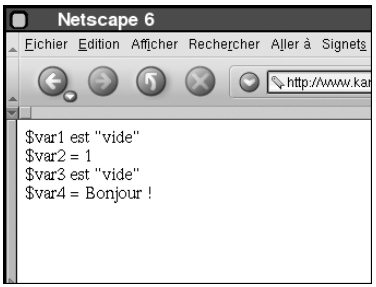


Figure 3.3 :
Le résultat dans un navigateur

Pour détruire une variable, utilisez la fonction `unset()`.

Afin de tester le type des variables, PHP possède une série de fonctions très commodes : `is_bool()`, `is_int()`, `is_double()`, `is_numeric()`, `is_string()`, `is_array()`, `is_scalar()`, `is_object()`, `is_resource()`, `is_NULL()`.

Toutes ces fonctions s'utilisent de la même manière. Elles renvoient `TRUE` si le type de la variable passée en paramètre est exactement le type recherché, et `FALSE` dans le cas contraire.

is_bool()

Détermine si une variable est de type booléen.

Syntaxe	`boolean is_bool(mixed $variable)`
`$variable`	Variable dont on veut déterminer si elle est de type booléen.
`retour`	`TRUE` si la variable est de type booléen, `FALSE` sinon.

```
<?php
$var = 1;
echo is_bool($var);
// Ne renvoie rien, le résultat du test est faux

$var = TRUE;
echo is_bool($var);
// Renvoie 1, le résultat du test est vrai
?>
```

is_int()

Détermine si une variable est de type entier.

Syntaxe	`boolean is_int(mixed $variable)`
`$variable`	Variable dont on veut déterminer si elle est de type entier.
`retour`	`TRUE` si la variable est de type entier, `FALSE` sinon.

`is_int()` est en fait un alias de `is_long()` dont `is_integer()` est un autre alias.

```
<?php
$var = "15";
echo is_int($var);
// Ne renvoie rien, le résultat du test est faux

$var = 15;
echo is_int($var);
// Renvoie 1, le résultat du test est vrai
?>
```


is_double()

Détermine si une variable est de type réel (décimal).

Syntaxe	`boolean is_double(mixed $variable)`
`$variable`	Variable dont on veut déterminer si elle est de type réel.
`retour`	`TRUE` si la variable est de type réel, `FALSE` sinon.

`is_float()` et `is_real()` sont des alias de la fonction `is_double()`.

```
<?php
$var = 15;
echo is_double($var);
// Ne renvoie rien, le résultat du test est faux

$var = 15.0;
echo is_double($var);
// Renvoie 1, le résultat du test est vrai
?>
```

is_numeric()

Détermine si une variable est un nombre ou une chaîne de caractères représentant un nombre.

Syntaxe	`boolean is_numeric(mixed $variable)`
`$variable`	Variable dont on veut déterminer si elle est de type nombre ou chaîne de caractères représentant un nombre.
`retour`	`TRUE` si la variable est de type nombre ou chaîne de caractères représentant un nombre, `FALSE` sinon.

```
<?php
$var1 = "quinze";
echo is_numeric($var1);
```

```
// Ne renvoie rien, le résultat du test est faux

$var2 = 15;
echo is_numeric($var2);
// Renvoie 1, le résultat du test est vrai

$var3 = "15";
echo is_numeric($var3);
// Renvoie 1, le résultat du test est vrai

$var4 = 15.0;
echo is_numeric($var4);
// Renvoie 1, le résultat du test est vrai
?>
```


is_string()

Détermine si une variable est de type chaîne de caractères.

Syntaxe	`boolean is_string( mixed $variable)`
`$variable`	Variable dont on veut déterminer si elle est de type chaîne de caractères.
`retour`	`TRUE` si la variable est de type chaîne de caractères, `FALSE` sinon.

```
<?php
$var = "15";
echo is_string($var);
// Renvoie 1, le résultat du test est vrai

$var = 15;
echo is_string($var);
// Ne renvoie rien, le résultat du test est faux
?>
```

is_array()

Détermine si une variable est de type tableau.

Syntaxe	`boolean is_array(mixed $variable)`
`$variable`	Variable dont on veut déterminer si elle est de type tableau.
`retour`	`TRUE` si la variable est de type tableau, `FALSE` sinon.

```
<?php
$var1 = 15;
echo is_array($var1);
// Ne renvoie rien, le résultat du test est faux
```

```
$var2 = array(15);
echo is_array($var2);
// Renvoie 1, le résultat du test est vrai
?>
```

is_object()

Détermine si une variable est de type objet.

Syntaxe	`boolean is_object(mixed $variable)`
`$variable`	Variable dont on veut déterminer si elle est de type objet.
`retour`	`TRUE` si la variable est de type objet, `FALSE` sinon.

```
<?php
class Bible {
var $x;
}

$var = new Bible();
$var->x = "Je suis un attribut";

echo is_object($var);
// Renvoie 1, le résultat du test est vrai

echo is_object($var->x);
// Ne renvoie rien, le résultat du test est faux
?>
```

is_scalar()

Détermine si une variable est de type scalaire.

Syntaxe	`boolean is_scalar(mixed $variable)`
`$variable`	Variable dont on veut déterminer si elle est de type scalaire.
`retour`	`TRUE` si la variable est de type scalaire, `FALSE` sinon.

```
<?php
$var1[0] = 3.14;
$var2 = 3.14;
$var3 = "pi";

echo is_scalar($var1);
// Ne renvoie rien, le résultat du test est faux
```

```
echo is_scalar($var1[0]);
// Renvoie 1, le résultat du test est vrai

echo is_scalar ($var2);
// Renvoie 1, le résultat du test est vrai

echo is_scalar ($var3);
// Renvoie 1, le résultat du test est vrai
?>
```


is_resource()

Détermine si une variable est de type resource.

Syntaxe	`boolean is_resource(mixed $variable)`
`$variable`	Variable dont on veut déterminer si elle est de type resource.
`retour`	`TRUE` si la variable est de type resource, `FALSE` sinon.

is_NULL()

Indique si une variable est `NULL`.

Syntaxe :	`boolean is_NULL(mixed $variable)`
`$variable`	Variable dont on veut déterminer si elle est `NULL`.
`retour`	`TRUE` si la variable est `NULL`, `FALSE` sinon.

```
<?php
$var = 0;
echo is_NULL($var);
// Ne renvoie rien, le résultat du test est faux

unset($var);
echo is_NULL($var);
// Renvoie 1, le résultat du test est vrai
?>
```

Les variables externes

Nous pouvons différencier deux types de variables externes :

- Les variables d'environnement (du serveur) ;
- Les variables passées en paramètre lors de l'appel de la page (soit via un formulaire soit via l'URL).

Les variables d'environnement

Les variables d'environnement sont des variables générées automatiquement par PHP à partir des données récupérées sur le serveur ou de l'en-tête des requêtes du client.


RENVOI

Voir le chapitre "Les en-têtes" pour plus de détails sur les en-têtes.

Ces variables constituent en fait deux tableaux de portée globale (accessibles depuis n'importe quel endroit du script) :

- `$_ENV` contient les "véritables" variables d'environnement du serveur, c'est-à-dire toutes les variables dont a pu hériter le compte sous lequel tourne le serveur web (ex. : PATH).
- `$_SERVER` contient, quant à lui, les variables "internes" du serveur (nom et version du serveur, adresse IP, etc.) ainsi que celles récupérées du client (adresse IP du client, type du navigateur, etc.).

Vous pouvez simplement visualiser toutes les variables d'environnement sur la page générée par la fonction `phpinfo()`.

Une autre possibilité pour récupérer la liste des variables est d'utiliser la fonction `get_defined_vars()`; elle permet de récupérer dans un tableau la liste de toutes les variables définies. Le tableau retourné contient alors toutes les variables d'environnement, mais aussi toutes les variables définies par l'utilisateur.

get_defined_vars()

Liste toutes les variables définies.

Syntaxe	`array get_defined_vars(void)`
`retour`	Toutes les variables dans un tableau associatif où les noms des clés sont les noms des variables et les valeurs celles des variables.

L'exemple ci-dessous permet de générer un tableau HTML contenant la liste des variables et leurs valeurs au moment de l'exécution du code :

```
<html>
<body>
<?php
$a1 = "première variable";
$a2 = "seconde variable";

echo "<table border=1>";
$a = get_defined_vars();
while (list($key, $val) = each($a)) {
  echo "<tr>";
```

```
  echo "<td>\$".$key."</td><td>".$val."</td>";
  echo "</tr>";
      }
echo "</table>";
?>
</body>
</html>
```

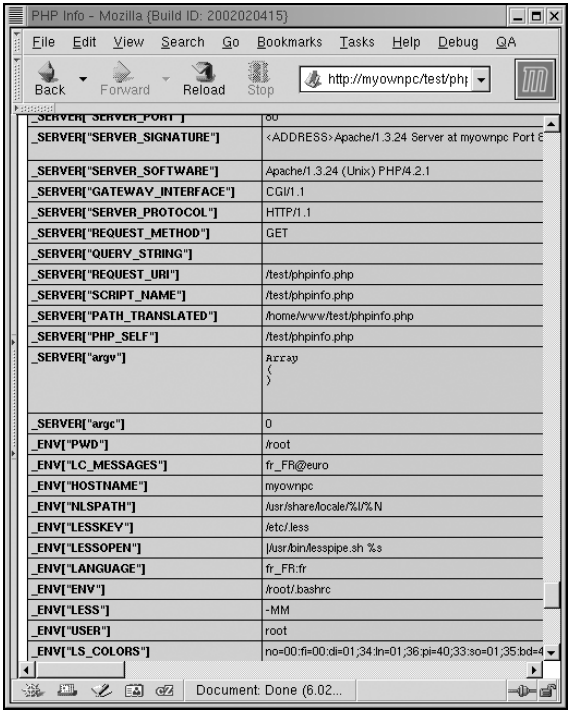


Figure 3.4 : *Ce code retourne un tableau contenant noms des variables et valeurs*

Le tableau `$_ENV` ne présente généralement pas un grand intérêt hormis dans quelques cas particuliers.

Vous pouvez aussi récupérer les valeurs à l'aide de la fonction `getEnv()`.

getEnv()

Retourne la valeur d'une variable d'environnement.

Syntaxe	`string getEnv(string $variableEnvironnement )`
`$variable Environnement`	Variable d'environnement dont on veut déterminer la valeur.
`retour`	La valeur de la variable d'environnement ou `FALSE` si il y a une erreur

Il est très simple de modifier les variables d'environnement en utilisant la fonction `putEnv()`. Les valeurs ainsi fixées par le développeur ne durent que le temps de l'exécution du script et, dès la fin de celui-ci, l'environnement par défaut est restauré.

Mais, comme la modification de certaines valeurs peut entraîner des problèmes de sécurité, l'administrateur du système a la possibilité d'empêcher certaines modifications. C'est d'ailleurs, par défaut, le cas de la variable `LD_LIBRARY_PATH` (d'un système UNIX/Linux) qui, si elle venait à être modifiée, pourrait éventuellement permettre aux pirates de masquer la présence d'une bibliothèque (peut-être utilisée par le serveur web) au profit d'une autre plus malicieuse. Ainsi, la directive `safe_mode_protected_env_vars` du fichier de configuration *php.ini* contient une liste de variables d'environnement que le développeur ne peut modifier. De la même façon, si `safe_mode` (mode protégé) est activé (ce qui n'est pas le cas par défaut), seules les variables commençant par les préfixes précisés dans `safe_mode_allowed_env_vars` pourront être modifiées.


RENVOI

Pour plus d'informations sur le fichier php.ini, reportez-vous au chapitre "Prise en main".

putEnv()

Fixe une nouvelle valeur à la variable d'environnement.

Syntaxe `void putEnv(string $chaineAffectation)`

`$chaineAffectation` Chaîne de caractères de la forme "<nom de variable>=<valeur>"

```
<?php
$ip = getEnv("REMOTE_ADDR");
// retourne l'adresse IP de l'utilisateur

putEnv("NOUV_IP=127.0.0.1");
// Définit une nouvelle variable d'environnement

echo getEnv("REMOTE_ADDR");
// Affiche 127.0.0.1
?>
```

Le tableau `$_SERVER`, quant à lui, va nous permettre de réaliser de nombreuses opérations.


REMARQUE

Variantes d'un serveur à l'autre
Les variables `$_SERVER` présentées sont celles disponibles sous Apache. La plupart sont également disponibles sur les serveurs IIS et iPlanet, mais vous observerez tout de même des différences.

Déterminer l'adresse IP du client

L'élément `$_SERVER["REMOTE_ADDR"]` retourne l'adresse IP du client, ce qui est particulièrement utile pour ne pas compter l'utilisateur plusieurs fois dans le nombre des visiteurs d'un site.

IP mouvante

La grande majorité des internautes n'est pas connectée en permanence à l'internet (même les utilisateurs de l'ADSL sont généralement contraints de se reconnecter au moins une fois par jour). Par conséquent, l'adresse IP (attribuée par le fournisseur d'accès) varie d'un jour sur l'autre. Il serait donc faux de supposer que deux visites à deux jours d'intervalle avec la même adresse IP corresponde au même visiteur. Alors que si l'intervalle de temps entre les deux visites est bien plus court, l'hypothèse est fort valable.

Déterminer d'où vient le client

Il est généralement possible de savoir quel lien a suivi le visiteur pour arriver sur le script en cours d'exécution. Pour cela, il suffit de lire le contenu de `$_SERVER["HTTP_REFERER"]` ; ceci retourne alors une URL.

Déterminer le type de navigateur du client

L'élément `$_SERVER["HTTP_USER_AGENT"]` permet de récupérer la chaîne d'identification fournie par le navigateur du client. Cela est particulièrement utile si vous avez à écrire des scripts générant du code Javascript un peu pointu, car, comme vous le savez peut-être, le Javascript dépend fortement du navigateur. Il en est de même de certaines balises HTML.

La valeur ainsi récupérée nécessite toutefois une petite analyse avant de déterminer avec précision la nature du navigateur, comme le démontre l'échantillon de valeurs possibles suivant :

- pour Internet Explorer :

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Mozilla/4.0 (compatible; MSIE 6.0; Windows 98)
Mozilla/4.0 (compatible; MSIE 5.0; Mac_PowerPC)
```

- Konqueror :

```
Mozilla/5.0 (compatibles; Konqueror/2.2.1; Linux)
```

- Netscape :

```
Mozilla/5.0 (Windows; U; Windows NT 5.0; en-US; rv:0.9.2) Gecko/20010726
✂ Netscape6/6.1
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:0.9.8) Gecko/20020204
```

- Opera :

```
Mozilla/4.7 [tr] (X11; I; Linux 2.2.16 i686)
```

```
Mozilla/4.0 (compatible; MSIE 5.0; Linux) Opera 6.0 [en]
```

Et tout cela se décline pour ainsi dire à l'infini. D'autant que certains visiteurs ne sont pas des utilisateurs derrière leur navigateur, mais des aspirateurs de sites (soit pour les moteurs de recherche, soit pour un particulier). Même si cela n'est pas systématique, la plupart précisent leur propre identifiant et d'autres offrent même la possibilité à l'utilisateur de définir sa propre chaîne d'identification. Voici, donc, un nouvel échantillon de valeur rencontrée :

- Les aspirateurs de sites et autres "spiders" de moteurs de recherche :

```
Googlebot/2.1 (+http://www.googlebot.com/bot.html)
Mozilla/3.0 (Slurp/si; slurp@inktomi.com; http://www.inktomi.com/slurp.html)
WebCopier v3.0
```

```
<html>
<body>
    Vous avez l'adresse IP :
    <?php echo $_SERVER["REMOTE_ADDR"]; ?>
  <br />
    Vous avez un navigateur qui porte la signature :
    <?php echo $_SERVER["HTTP_USER_AGENT"]; ?>
  <br />
</body>
</html>
```

Déterminer la langue du visiteur

Bien que cette fonctionnalité ne soit pas forcément bien connue, les navigateurs offrent la possibilité de déterminer ses langues préférées. Il est alors possible, lorsque l'on gère un site multi-langue, de s'adapter automatiquement en proposant la langue qui répondra au mieux aux attentes du visiteur.

Ce paramètre est accessible depuis le serveur via la variable `$_SERVER["HTTP_ACCEPT_LANGUAGE"]` qui aura une valeur du genre :

```
fr, en;q=0.5
```

Il s'agit donc d'une chaîne contenant les codes ISO des langues, séparés par des virgules et par ordre de préférence. La liste se termine par un point-virgule (et l'on trouve ensuite un coefficient).

Déterminer le nom du serveur web

Il est possible de déterminer le nom du serveur en consultant le contenu de `$_SERVER["SERVER_NAME"]`, ce qui peut vous permettre de créer un script unique ayant un comportement différent selon la machine sur lequel il tourne.

Déterminer la racine du serveur web

Si vous avez à manipuler des noms de fichier, le chemin relatif à la racine du site web ne sera peut-être pas toujours suffisant. Si vous avez besoin du chemin complet depuis la racine du disque dur, vous pourrez faire appel à `$_SERVER["DOCUMENT_ROOT"]` pour récupérer le chemin depuis la racine du disque dur vers la racine du serveur web.


REMARQUE

iPlanet, IIS et PWS
Cette variable n'est pas disponible sur les serveurs iPlanet, IIS et PWS.

Déterminer le nom du script en cours d'exécution

La variable `$_SERVER["PHP_SELF"]` contient le chemin absolu (commence par un `/`) par rapport à la racine du site web et le nom du script en cours d'exécution. Cela est particulièrement intéressant lorsqu'il s'agit d'écrire des scripts qui doivent s'appeler eux-mêmes (ce qui est souvent le cas des scripts générant des formulaires) et que l'on ne souhaite pas indiquer "en dur" le nom du script (pour ne pas avoir de soucis dans le cas où celui-ci serait renommé). Cela peut également servir pour déterminer le chemin.

La variable `$_SERVER["SCRIPT_FILENAME"]`, quant à elle, donne le chemin absolu par rapport à la racine du disque dur.


ATTENTION

Exécution !
Le script en cours d'exécution n'est pas nécessairement le script dans lequel est écrite l'instruction en cours d'exécution. Si un script A inclut un script B au niveau du script B, `$_SERVER["PHP_SELF"]` et `$_SERVER["SCRIPT_FILENAME"]` retourneront le chemin du script A. Pour connaître le chemin du script B, il faudra faire appel à la constante `__FILE__` qui donnera le chemin absolu par rapport à la racine du disque dur.


REMARQUE

iPlanet, IIS et PWS
La valeur `$_SERVER["SCRIPT_FILENAME"]` n'est pas disponible sous iPlanet, IIS et PWS ; à la place, vous pouvez utiliser `$_SERVER["PATH_TRANSLATED"]`.

Autres...

Même si nous avons vu ici les principales variables d'environnement, il en existe de nombreuses autres.


RENVOI

Vous pouvez vous reporter aux annexes de ce livre pour une liste plus complète des variables d'environnement.


REMARQUE

Souvenez-vous...
Avant PHP 4.1.0, les tableaux `$_SERVER`, `$_ENV` s'appelaient `$HTTP_SERVER_VARS`, `$HTTP_ENV_VARS`, etc.

Les variables passées en paramètre du script (ou via des formulaires)

Comme vous le savez peut-être déjà (en tout cas, vous allez bientôt le savoir) il est possible de passer des paramètres à un script.

Il existe différentes façons de passer ces paramètres, mais la plus simple - ou du moins la plus parlante - consiste à ajouter les paramètres et leurs valeurs à la suite de l'URL après un point d'interrogation '?' chaque couple <nom du parametre>=<valeur> étant séparé d'un autre par le caractère "et commercial" '&'. On parle alors de passage de paramètres par la méthode GET.

Ce qui donne une URL du genre :

http://mondomaine.com/monscript.php?nom=François&prenom=DUPOND

Dans ce cas, les différents paramètres du script seront disponibles dans un tableau appelé $_GET.

Ainsi, si vous appelez le script suivant de la manière indiquée précédemment,

Listing 3.2 : get_echo.php

```
<?php
echo "Nom =".$_GET["nom"]."<br />";
echo "Prénom =". $_GET["prenom"]."<br />";
?>
```

vous obtiendrez le résultat suivant :

```
Nom = François
Prénom = DUPOND
```

Je pense que vous commencez à entrevoir des possibilités d'applications. Mais vous serez plus enthousiasmé encore après avoir vu comment utiliser des formulaires de saisie.

Rappel sur les formulaires

Sans faire un descriptif complet des balises HTML, voici un rapide aperçu des balises généralement utilisées pour créer un formulaire.

Pour commencer, la définition d'un formulaire se fait entre des balises <form> et </form>.

La balise <form> possède, entre autres, deux attributs importants :

- Un attribut action précisant quel page doit être chargée lorsque le formulaire sera validé.
- Un attribut method précisant quel mode d'envoi des données doit être utilisé. Ce dernier peut prendre l'une des deux valeurs suivantes get (que nous venons de voir) ou post (que nous n'allons pas tarder à voir).

La plupart des éléments d'un formulaire se construisent avec la balise <input />.

La balise <input> possède, entre autres, trois attributs importants :

- Un attribut type précisant le type de champ de saisie (voir ci-après).

- Un attribut `name` précisant le nom du paramètre associé à ce champ.
- Un attribut `default` ou `checked` (selon les cas) précisant la valeur par défaut.

L'attribut `type` peut prendre les valeurs :

- `text` pour préciser un champ de saisie texte.
- `radio` pour préciser un élément d'une liste de sélection dans laquelle une seule des options peut-être sélectionnée.
- `checkbox` pour préciser une case à cocher.

Enfin, vous disposez également des balises `<select>` (associée à `<option>`) et `<textarea>`.

Application

Voici donc un petit exemple d'application :

Listing 3.3 : form_get.html

```
<html>
<body>
<form action="get_echo.php" method="get">
Veuillez indiquer vos Noms et Prénoms<br />
Prénom: <input type="text" name="prenom" /><br />
Nom: <input type="text" name="nom" /><br />
<input type="submit" />
</form>
</body>
</html>
```

Comme vous pouvez le constater dans la barre d'adresse de votre navigateur, le simple fait de valider ce formulaire appelle le script précisé dans le champ action (ici, le script `get_echo.php` présenté précédemment) en ajoutant à l'URL la liste des paramètres et de leurs valeurs, comme cela peut se faire avec une méthode `GET` rédigée manuellement.


REMARQUE

Sélection multiple

Si vous souhaitez utiliser un élément de formulaire pouvant retourner une sélection multiple (typiquement une balise `<select>` en mode "multiselect" ou encore une série de cases à cocher portant toutes le même nom), vous devez alors lui affecter un nom de tableau (i.e. un nom suivi de `[]`). La valeur récupérée dans `$_GET` ne sera alors pas de type `string` mais de type `array`.

Le fait d'utiliser la méthode `GET` peut entraîner des problèmes d'ordres divers. Le problème le plus évident est lié à la longueur de l'URL. Plus il y a de paramètres à passer et plus l'URL sera longue, ce qui pourrait mettre en défaut votre serveur. Un autre problème, plus subtil (mais non des moindres) concerne la sécurité. En effet, si vous utilisez une méthode `GET` pour passer des mots de passe (pour une section membre par exemple), ceux-ci apparaîtront dans l'URL de la page. Outre le fait que le mot de passe devienne visible dans la barre d'adresse du navigateur

et dans l'historique, le problème s'aggrave quand, depuis cette page, vous proposez un lien vers un autre site (site d'une tierce personne). En effet, si ce site établit des statistiques sur ses consultations, peut-être trace-t-il l'information `$_SERVER["HTTP_REFERER"]` qui note d'où vient le visiteur. Dans ce cas, l'administrateur de ce site pourra lire l'URL complète, indiquant l'adresse de votre site et un mot de passe généralement accompagné d'un nom d'utilisateur ! Bref, il aura en main tous les outils nécessaires pour accéder à des informations qui ne lui sont pas destinées…

Pour pallier ces différents problèmes, vous avez la possibilité d'utiliser la méthode `POST`. Dans ce cas, les paramètres ne sont pas ajoutés à l'URL, mais passés "discrètement" dans l'en-tête de la requête HTTP envoyée au serveur.

Dans ce cas, les deux fichiers présentés précédemment deviennent :

Listing 3.4 : form_post.html

```
<html>
<body>
<form action="get_echo.php" method="post">
Veuillez indiquer vos Noms et Prénoms<br />
Prénom: <input type="text" name="prenom" /><br />
Nom: <input type="text" name="nom" /><br />
<input type="submit" />
</form>
</body>
</html>
```

Listing 3.5 : post_echo.php

```
<?php
echo "Nom =".$_POST["nom"]."<br />";
echo "Prénom =". $_POST["prenom"]."<br />";
?>
```

Vous noterez donc que le travail du développeur est identique, puisqu'il suffit de remplacer `get` par `post`. Ainsi, le tableau contenant les données s'appelle `$_POST`.

En dehors de l'utilisation de formulaires, la méthode `GET` reste la seule solution facilement implémentable pour passer des paramètres à un script.


REMARQUE

Souvenez-vous...

Avant PHP 4.1.0, les tableaux `$_GET`, `$_POST` s'appelaient `$HTTP_GET_VARS`, `$HTTP_POST_VAR`.

Avant PHP 4.2.0, le fichier de configuration php.ini fixait par défaut l'option `register_global` à `on`, ce qui avait pour effet de créer systématiquement une variable globale portant le nom des paramètres `GET`, `POST`, etc. Ainsi, par exemple, `$_GET["nom"]` était également accessible par `$nom`, ce qui engendrait des problèmes de sécurité. Cela est particulièrement vrai avec les fichiers destinés à être inclus et qui s'appuient sur des variables globales déclarées dans le script appelant.

Car, dans ce cas, une variable `$maVariable` *peut être "piratée" en appelant le script inclus avec le paramètre* `?maVariable=autreValeur`.

Vous pouvez vous reporter au chapitre "Prise en main" pour plus de détails sur le fichier php.ini.

Il existe également d'autres variables d'environnement, mais nous leur consacrerons un chapitre spécifique.

Il s'agit des tableaux `$_COOKIE` et `$_SESSION` étudiés dans le chapitre *En-têtes* et du tableau `$_FILES` étudié dans le chapitre *Fichiers*.

3.5. Les opérateurs

Arithmétiques

Les opérateurs mathématiques sont classiques ; les voici regroupés dans un tableau. Ils agissent aussi bien sur des entiers que sur des réels.

Tableau 3.6 : Les opérateurs arithmétiques

Opérateur	Description
`$a + $b`	Somme de `$a` et `$b`.
`$a - $b`	Différence de `$a` et `$b`.
`$a * $b`	Produit de `$a` par `$b`.
`$a / $b`	Quotient de `$a` par `$b`.
`$a % $b`	Reste de la division de `$a` par `$b`.

L'opérateur '`/`' renvoie un entier si les deux valeurs de la division sont des entiers ou des chaînes de caractères représentant des entiers. Si une des deux valeurs est un réel, alors le résultat sera un réel.

Reste de la division
L'opérateur '`%`*' est particulièrement utile ; il peut permettre de savoir si un nombre est divisible par un autre. Par exemple, si vous souhaitez savoir si* `$a` *est un chiffre pair, il suffit de tester le résultat de* `$a%2` *s'il vaut 0 alors* `$a` *est un chiffre pair (divisible par 2).*

```
<?php
  for ($i=1; $i<= 5; $i++) {
    echo "$i:";
    if ($i%2 == 0) echo "impair "; else echo "pair ";
```

```
  }
?>
```

Le résultat :

```
1:impair 2:pair 3:impair 4:pair 5:impair
```

Binaires

Vous pouvez traiter les données binaires à l'aide de ces fonctions primaires :

Tableau 3.7 : Les opérateurs binaires

Exemple	Nom	Description
`$a & $b`	Et	Active les bits présents dans `$a` et `$b/`.
`$a | $b`	Ou	Active les bits présents dans `$a` ou `$b`.
`$a ^ $b`	Ou exclusif	Active les bits présents dans `$a` ou `$b` mais pas dans les deux.
`~$a`	Non	Oppose les bits.
`$a<<$b`	Décalage à gauche	Décale `$a` à gauche de `$b` rangs (revient à multiplier `$b` fois par 2).
`$a>>$b`	Décalage à droite	Décale `$a` à droite de `$b` rangs (revient à diviser `$b` fois par 2).

Considérons `$a=10` soit 1010 en binaire, que l'on notera 1010b et `$b=13` soit 1101b

```
<?php
$a=10;
$b=13;
echo "$a & $b = ".($a&$b);
echo "$a | $b = ".($a|$b);
echo "$a ^ $b = ".($a ^ $b);
echo "~$a = ".(~$a);
echo "$a<<2 = ".($a<<2);
echo "$a>>2 = ".($a>>2);
?>
```

Voici le résultat obtenu :

Figure 3.5 : *Résultat*

Vérifions :

- 1010b AND 1101b fait 1000b soit 8
- 1010b OR 1101b fait 1111b soit 15
- 1010b XOR 1101b fait 0111b soit 7
- NOT 1010b fait …10101b soit en complément a 1 1011b donc -11
- 1010b<<2 fait 101000 soit 40
- 1010b>>2 fait 10 soit 2

Les résultats concordent bien avec nos calculs.

L'écriture binaire
Si vous n'êtes pas familier de la notation et des calculs binaires, cela peut vous paraître obscur. Bien que cela ne soit pas indispensable pour l'apprentissage de PHP, si vous voulez en apprendre davantage sur la notation binaire, vous pouvez vous connecter, par exemple, aux sites Internet suivants :
http://www.histoire-informatique.org/technologie/binaire.html
http://www.web2.cnam.fr/evariste/evariste/10_cours/binaire/binaire.htm
http://www.lille.iufm.fr/labo/pagesProjets/leparc/base2/thema.htm

Chaînes de caractères

L'opérateur permettant la concaténation (fusion) de deux chaînes de caractères et le point '.'.

Voici un exemple de script affichant le résultat de la concaténation de deux chaînes :

```
<?php
    $a="PHP ?";
    $b=" Facile !";
    echo $a.$b;
?>
```

Affectation

Le signe = est le symbole utilisé pour affecter une valeur à une variable ; la valeur à la droite de ce signe est affectée à la variable de gauche. Il est possible de les mettre en séquence, comme dans le script qui suit :

```
<?php
$a = ( $b = 3 ) +5;
?>
```

Après cette instruction, `$a` vaut 8 et `$b` vaut 3.

Il existe des raccourcis particulièrement utiles : au lieu d'écrire `$a = $a+10`, vous pouvez écrire `$a += 10`. De même, `$a = $a-10` peut s'écrire `$a -= 10`. `/=` et `*=` sont aussi des opérateurs d'affectation.

Pour les chaînes de caractères, il existe le même type d'opérateur, à savoir : `.=`, écrire `$a .= "toto";` est équivalent à `$a = $a."toto";`

Incrémentation et décrémentation

Tableau 3.8 : Opérateurs d'incrémentation et de décrémentation

Syntaxe	Nom	Description
`++$a`	Pré-incrémentation	Incrémente `$a` de un puis retourne `$a`.
`$a++`	Post-incrémentation	Retourne `$a` puis incrémente `$a` de un.
`--$a`	Pré-décrémentation	Décrémente `$a` de un puis retourne `$a`.
`$a--`	Post-décrémentation	Retourne `$a` puis décrémente `$a` de un.

```
<?php
    echo "Post-incrementation<br>";
    $a = 5;
    echo $a++;  // affiche 5
    echo "<br>";// place une nouvelle ligne dans le script resultat
    echo $a;    // affiche 6
    echo "<br>";
    echo "Pre-incrementation<br>";
    $a = 5;
    echo ++$a;   // affiche 6
    echo "<br>";
    echo $a;     // affiche 6
    echo "<br>";

    echo "Post-decrementation<br>";
    $a = 5;
    echo $a--;  // affiche 5
    echo "<br>";
    echo $a;    // affiche 4
    echo "<br>";

    echo "Pre-decrementation<br>";
    $a = 5;
    echo --$a;   // affiche 4
    echo "<br>";
    echo $a;    // affiche 4
    echo "<br>";
?>
```

Comparaison

Tableau 3.9 : Les opérateurs de comparaison

Syntaxe	Nom	Description
`$a == $b`	Égal	Renvoie `TRUE` si `$a` est égal à `$b`.
`$a === $b`	Identique	Renvoie `TRUE` si `$a` est égal à `$b` et que `$a` et `$b` sont du même type.
`$a != $b`	Différent	Renvoie `TRUE` si `$a` est différent de `$b`.
`$a <> $b`	Différent	Renvoie `TRUE` si `$a` est différent de `$b`.
`$a !== $b`	Non identique	Renvoie `TRUE` si `$a` est différent de `$b` ou de type différent.
`$a < $b`	Inférieur à	Renvoie `TRUE` si `$a` est inférieur à `$b`.
`$a > $b`	Supérieur à	Renvoie `TRUE` si `$a` est supérieur à `$b`.
`$a <= $b`	Inférieur ou égal à	Renvoie `TRUE` si `$a` est inférieur ou égal à `$b`.
`$a >= $b`	Supérieur ou égal à	Renvoi `TRUE` si `$a` est supérieur ou égal à `$b`.

Tous ces comparateurs vous permettront de tester les variables pour agir en fonction du résultat obtenu.


REMARQUE

Typage

Depuis PHP4, le typage est devenu plus fort, même si l'on peut encore comparer directement des chaînes de caractères avec un entier ou un réel par exemple. Mais si `FALSE == 0` *(ce qui peut porter à confusion lorsqu'une fonction peut retourner* `0` *dans un cas nominal et* `FALSE` *en cas d'erreur) et bien* `FALSE !== 0` *(puisque* `FALSE` *est un booléen et* `0` *un entier).*

Bien entendu, il est possible de combiner les opérateurs binaires et les opérateurs de comparaison.

L'expression correspondant aux phrases "la variable `a` est différente de 3 et la variable `b` vaut toto ou alors la variable `c` est un booléen valant `FALSE`" s'écrit de différentes façons. En voici quelques-unes :

```
(($a != 3 && $b == "toto") || ($c==FALSE))
// faire un test d'égalité avec TRUE n'est pas très futé
(($a != 3 && $b == "toto") || (!$c==TRUE))
// la façon la plus standard de faire, reste
(($a != 3 && $b == "toto") || (!$c))
```

Logique

Les opérateurs de logique vont servir à faire des tests complexes, par exemple pour savoir si `$a` vaut 3 ET `$b` est faux.

Dans le tableau suivant sont regroupés tous les opérateurs logiques supportés par PHP :

Tableau 3.10 : Les opérateurs logiques

Syntaxe	Description
`$a and $b`	VRAI si `$a` et `$b` sont vrais.
`$a && $b`	VRAI si `$a` et `$b` sont vrais.
`$a or $b`	VRAI si `$a` ou `$b` est vrai.
`$a || $b`	VRAI si `$a` ou `$b` est vrai.
`$a xor $b`	VRAI si `$a` ou `$b` est vrai, mais pas les deux.
`! $a`	VRAI si `$a` est faux.

Utiliser l'opérateur OR comme expression conditionnelle
PHP n'interprète pas la partie droite de l'expression `OR` *si la partie gauche est vraie. On peut ainsi s'en servir astucieusement pour exécuter une instruction si une expression est incorrecte.*

```
<?php
$a=4;
($a==3) OR die("\$a ne vaut pas 3");
?>
```

Ce script stoppe si `$a` *ne vaut pas 3 et affiche un message.*
Cela est particulièrement utile lors de l'appel à des fonctions (par exemple de connexion à une base de données) qui retournent `FALSE` *en cas d'erreur.*

```
<?php
connexion() OR die("Impossible de se connecter");
?>
```

Contrôles d'erreur

PHP intègre quatre grands niveaux de message d'erreur ou d'alerte. À chacun correspond une constante :

- `E_NOTICE` : simple remarque (comme lorsque l'on essaye d'accéder à un index inexistant d'un tableau).
- `E_PARSE` : en cas d'erreur d'analyse à la compilation (comme une erreur de syntaxe).

- `E_WARNING` : une alerte ne nécessitant pas l'arrêt du script.
- `E_ERROR` : une erreur "grave" nécessitant l'arrêt du script.

Une constante supplémentaire fait la somme de tous ces types d'erreur et d'alerte, il s'agit de `E_ALL`.

À ceux-là viennent s'ajouter les niveaux `E_CORE_ERROR`, `E_CORE_WARNING`, `E_COMPILE_ERROR`, `E_COMPILE_WARNING`, `E_USER_ERROR`, `E_USER_WARNING` et `E_USER_NOTICE` d'une utilisation moins courante.

Lorsqu'une erreur est rencontrée et qu'elle dépasse le seuil d'alerte, ou plus exactement lorsque son niveau de sévérité fait partie de la liste des niveaux à signaler, un message est envoyé au client (i.e. est affiché sur la page reçue par le navigateur).

Les niveaux d'erreur à signaler sont définis dans le fichier de configuration *php.ini*, mais peuvent être modifiés au niveau du script par un simple appel à la fonction `error_reporting()`.

error_reporting()

Fixe les niveaux d'erreur devant être signalés.

Syntaxe	`int error_reporting([int $niveaux])`
`$niveaux`	Combinaison par `OU` logique ou `OU` exclusif des différentes constantes indiquées précédemment.
`retour`	Anciennes valeurs des niveaux d'erreur.

Il est toutefois possible de s'affranchir des messages d'erreur pour une instruction donnée sans avoir recours à cette fonction. Pour cela, il existe un opérateur de contrôle d'erreur, c'est le caractère `@` ; il peut être mis devant n'importe quelle fonction susceptible de générer une erreur.

Cela peut être particulièrement utile pour générer vos propres messages d'erreur.

Le script suivant fixe les niveaux d'alerte à "tous sauf les simples remarques" (ce qui est la configuration par défaut de PHP), puis tente d'accéder à un fichier qui n'existe pas.

```
<?php
    error_reporting(E_ALL ^ E_NOTICE);
    $fichier = @file ('fichier_inexistant.toto') or
      die ("Impossible d'ouvrir le fichier.");
?>
```

La fonction `file` renvoie un booléen indiquant si l'opération s'est bien déroulée. Si le fichier n'existe pas, alors la fonction renvoie un message. Ici, nous avons mis `@` devant le nom de la fonction ; nous n'aurons donc pas cet affichage, mais le texte "Impossible d'ouvrir le fichier".

Notez que cela peut aussi s'appliquer à des expressions, et donc à un tableau, pour éviter un message si l'élément recherché n'existe pas (ex. : `@tableau["index_inexistant"];`). Ceci

dit, le message prévu en pareil cas est de niveau NOTICE, et PHP n'est, par défaut, pas configuré pour afficher ce type de message.

Connexion à une base de données
Lorsque vous vous connectez à une base de données, il se peut que celle-ci soit momentanément indisponible. Donc, au lieu d'obtenir le message d'erreur standard, il vaut mieux faire précéder la fonction de connexion de `@` *et tester le code retour pour afficher votre propre message si la connexion a échoué.*

Exécution

Il existe un opérateur d'exécution, c'est ″. Ce qui se trouve entre ces apostrophes inverses sera interprété par PHP comme une commande shell. Voici un exemple de script qui liste le contenu d'un répertoire, le stocke dans une variable, et l'affiche :

Listing 3.6 : version Unix/Linux

```
<?php
    $liste='ls –al';
    echo "<pre>$liste</pre>";
?>
```

Listing 3.7 : version Windows

```
<?php
    $liste='dir';
    echo "<pre>$liste</pre>";
?>
```

Configuration du serveur
Ceci ne fonctionne pas si l'option `safe_mode` *du fichier de configuration php.ini a été activée (*`on`*) ou si la fonction* `shell_exec()` *est désactivée.*

Vous pouvez vous reporter au chapitre "Prise en main" pour plus de détails sur le fichier php.ini.

Priorités

Les priorités sur les calculs sont les mêmes que celles que vous connaissez déjà.

`8 + 2 * 3` fera bien `8 + 6 = 14` (et pas `10 * 3 = 30`).

Dans le tableau suivant, vous trouverez les opérateurs par ordre de priorité ; les premiers éléments sont prioritaires sur les derniers.

Tableau 3.11 : Associativité et priorité des opérateurs par ordre décroissant de priorité

Opérateur	Associativité
`New`	Aucune
`[`	Droite
`! ~ ++ -- (int) (double) (string) (array) (object) @`	Droite
`* / %`	Gauche
`+ - .`	Gauche
`<< >>`	Gauche
`< <= > >=`	Aucune
`== != === !==`	Aucune
`&`	Gauche
`^`	Gauche
`|`	Gauche
`&&`	Gauche
`||`	Gauche
`? :`	Gauche
`= += -= *= /= .= %= &= |= ^= ~= <<= >>=`	Gauche
`,`	Gauche

3.6. Les structures de contrôle

Extrêmement utiles, les structures de contrôle vous permettront de faire des boucles et des tests dans vos scripts PHP.

If, else, elseif

La boucle de contrôle `if` est sans aucun doute la plus importante dans tous les langages de programmation ; cela vaut aussi en PHP.

La syntaxe est classique :

```
if (expression) instruction;
```

ou, s'il y a plusieurs instructions (et cela reste conseillé même s'il n'y a qu'une seule instruction) :

```
if (expression) {
    instruction1;
    instruction2;
    instruction3;
}
```

`expression` est une expression booléenne ; si elle est vérifiée (`TRUE`), la ou les expressions qui suivent sont évaluées :

```
<?php
    $a=3;
    if ($a>1) echo "a est supérieur à un";
?>
```

Expressions
Les expressions peuvent bien entendu utiliser tous les opérateurs de comparaison et de logique. Il faut toutefois s'assurer que les parenthèses sont correctement placées.

Maintenant, il est très probable que vous ayez envie d'exécuter certaines instructions si l'expression retourne `TRUE` (Vrai), et d'autres si elle retourne `FALSE` (Faux). Dans ce cas, il faut utiliser l'expression suivante :

```
if (expression) {
    instruction1;
    instruction2;
    instruction3;
} else {
    instruction4;
    instruction5;
}
```

Voici un exemple :

```
<?php
    $a=3;
    if ($a>1) echo "a est supérieur à un";
    else echo "a est inférieur ou égal à un";
?>
```

Pour obtenir une valeur ou une autre selon le résultat d'un test, il est possible d'utiliser les instructions `?` et `:` selon le schéma suivant :

```
(expression) ? (valeur1) : (valeur2)
```

Ce qui donne par exemple :

```
<?php
    $a=3;
    echo "a est ".(($a>1)?"supérieur":"inférieur ou égal")." à un";
.>
```


REMARQUE

Accolades
Encore une fois, dans le cas d'une seule instruction, les accolades peuvent être omises, mais cela est déconseillé (comme l'indiquent les règles de codage).

Il se peut que vous souhaitiez faire plusieurs tests consécutifs ; pour cela vous pouvez utiliser les instructions `if..elseif..else`…

```
if (expression) {
    instruction1;
    instruction2;
    instruction3;
} elseif {
    instruction4;
    instruction5;
} else {
    instruction6;
    instruction7;
}
```

Reprenons l'exemple précédent et améliorons-le :

```
<?php
    $a=3;
    if ($a>1) echo "a est supérieur à un";
    elseif ($a<1) echo "a est inférieur à un";
    else echo "a vaut 1";
?>
```


REMARQUE

Jouer l'alternance
Comme cela a déjà été évoqué, les portions de script PHP peuvent être placées comme bon vous semble dans le script (vous pouvez ainsi alterner les portions de PHP et de HTML).
Ainsi, le script suivant est valide :

```
<?php
    $a=3;
    if ($a>1) {
?>
    a est supérieur à un
<?php
    } elseif ($a<1) {
?>
    a est inférieur à un
<?php
    else {
?>
    a vaut 1
<?php
```

```
    }
?>
```

Il existe encore une autre façon (même si nous la déconseillons) de noter. Au lieu d'utiliser les accolades, la notation suivante est également valide. Dans ce cas, la fin du bloc est indiquée par `endif`.

```
if (expression):
    instruction1;
    instruction2;
    instruction3;
elseif:
    instruction4;
    instruction5;
else:
    instruction6;
    instruction7;
endif;
```

Exemple (première étape)

À partir de cette boucle de contrôle, il est déjà possible de faire des scripts intéressants.

Réalisons ensemble un script de Quizz.

Listing 3.8 : quizz01.php

```
<html>
<head>
  <title>Quizz</title>
</head>
<body>
<?php
if (isset($_POST["reponse"])) {
    $reponse=$ _POST["reponse"];
} else {
    $reponse="";
}
?>
<p>Quel est le site officiel de PHP ?</p>
  <form type="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
   <input type="text" name="reponse" value="<?php echo $reponse ?>" />
   <input type="submit" value="OK" />
  </form>
  <?php
  if ($reponse!="") {
      if (strtolower($reponse) == "www.phpfacile.com")
          echo "Ce n'est pas la bonne réponse";
     elseif (strtolower($reponse) == "www.php.net") echo "Bingo !";
     else echo "Et non, ce n'est pas ".$reponse;
  }
 ?>
```

```
</body>
</html>
```

Ici, la seule structure utilisée est `if elseif else`. Le fichier HTML de départ est un simple formulaire qui s'envoie à lui-même la réponse saisie.

Voici ce que signifie ce script :

Si la variable `$reponse` est définie, alors on compare la valeur de la variable réponse transformée en minuscules avec la chaîne "`www.phpfacile.com`".. Si ces chaînes sont identiques alors "Ce n'est pas la bonne réponse" est affiché, sinon on compare à "`www.php.net`" .. Si ces chaînes sont identiques "Bingo" est affiché, sinon la phrase "Eh non, ce n'est pas " suivie de la réponse entrée est affichée.

Figure 3.6 : *Quizz v0*

While, do … while

Pour répéter une série d'instructions, l'instruction `while` ou "tant que" en français vous sera utile. Voici la syntaxe :

```
while (expression) instruction;
```

Cela signifie que tant que l'expression sera VRAIE, l'instruction sera exécutée. Il faut donc s'assurer que l'expression passe obligatoirement à FAUX à un moment donné, sinon le script bouclera à l'infini et ne se terminera que lorsque le temps maximal d'exécution d'un script sera atteint.

Bien sûr, l'utilisation des accolades est également possible (et conseillée) ici.

```
while (expression) {
    instruction1;
    instruction2;
}
```

La notation avec les `:` est également valide (quoique déconseillée). Dans ce cas, le bloc est terminé par `endwhile`.

```
while (expression):
    instruction1;
    instruction2;
endwhile;
```

Voici un exemple concret de script qui affiche les entiers de 1 à 10.

Listing 3.9 : Exemple

```
<?php
    $i = 1;
    while ($i <= 10) {
        echo print $i++;
    }
?>
```

Durée de vie d'un script
Vous pouvez régler le temps d'exécution d'un script dans le fichier de configuration de PHP : il suffit de donner une valeur en secondes à `max_execution_time` *au niveau du fichier de configuration php.ini ou via la fonction* `set_time_limit()`*. Ainsi, s'il s'avère que votre boucle ne se termine pas, le script ne tournera pas indéfiniment.*

Vous pouvez vous reporter au chapitre "Prise en main" pour plus de détails sur le fichier php.ini.

Une variante de la boucle `while` est la boucle `do…while`. La différence réside dans ce que le test est fait après les instructions. Ainsi, la série d'instructions est obligatoirement effectuée au moins une fois avant d'être testée.

Voici la syntaxe :

```
do {
    instruction1;
    instruction2;
} while (expression);
```

For, foreach

Les boucles `for` sont très utilisées ; elles fonctionnent, comme dans d'autres langages, de la façon suivante :

```
for (expression1; expression2; expression3) instruction
```

`expression1` est exécutée avant toute itération. Elle sert donc généralement à initialiser les variables de la boucle.

`expression2` est exécutée à chaque itération. Si elle est VRAIE, une nouvelle exécution des instructions a lieu, sinon le programme sort de la boucle.

`expression3` est une instruction exécutée à chaque itération. Elle sert généralement à incrémenter une variable.

Voici un exemple de script qui affiche les chiffres de 0 à 9 :

```
for ($i=0;$i<10;$i++) echo $i;
```

Il faut lire cette commande comme suit : après avoir initialisé `$i` à 0, et tant que `$i` est inférieur à 10, afficher `$i`, puis l'incrémenter.

L'instruction `break` permet de sortir de la boucle. Dans ce cas, le restant des instructions n'est pas exécuté.

```
for ($i=0;;$i++) {
    echo $i;
    if ($i==9) break;
}
```

Ces lignes de code auront le même effet que le script précédent.

Il est aussi possible de se servir de la syntaxe utilisant les deux-points (:). Dans ce cas, le bloc se termine par `endfor`.

```
for (expression1;expression2;expression3):
    instruction1;
    instruction2;
endfor;
```

L'instruction `foreach`. offre une autre manière de réaliser une boucle, et elle est particulièrement pensée pour les tableaux. Voici deux syntaxes possibles :

```
foreach($tableau as $valeur) instruction;
foreach($tableau as $cle => $valeur) instruction;
```

Dans la boucle, la valeur de l'entrée du tableau est récupérable par `$valeur` ; la clé associée est récupérable par `$cle`, à condition d'utiliser la seconde syntaxe dans cet exemple.

Pointeur de tableau
Lorsque la boucle `foreach` est invoquée, elle met le pointeur du tableau en première position. Il est donc inutile d'appliquer la fonction `reset()` dessus.

Exemple (deuxième étape)

Cette fois, le script proposé permet de répondre à plusieurs questions sur une même page. Il utilise une boucle `foreach` et une boucle `for` ainsi que des tests `if`.

Listing 3.10 : quizz02.php

```
<html>
<head>
  <title>Quizz</title>
</head>
<body>
<?php
$questions = array("Quel est le site officiel de PHP ?",
                   "Quel est le nom de la commande qui
```

```
                    affiche une chaîne de caractère ?",
                   "Quel est le nom de la commande
                    qui met une chaîne de caractère en minuscule ?");
$reponses=array("www.php.net","echo","strtolower");
// initialisation des variables
foreach($questions as $index => $question) {
    if (isset($_POST["reponse$index"])) {
        ${"reponse$index"} = $_POST["reponse$index"];
    } else {
        ${"reponse$index"} = "";
    }
}
?>
  <form type="post" action="<?php echo $_SERVER["PHP_SELF"];?>">
  <?php
      foreach($questions as $index => $question) {
          echo $question;
          echo "<input type=\"text\" name=\"reponse$index\"
                value=\"${"reponse$index"}\"><br>";
  }
 ?>
  <input type="submit" value="OK">
  </form>
  <?php
      if ($reponse0 != "") {
          $parfait = TRUE;
          for ($i=0; $i<sizeof($questions); $i++) {
              if (strtolower(${"reponse$i"}) == $reponses[$i]) {
                  echo "<br />La réponse ".($i+1)." est correcte";
              } else {
                  echo "<br />La réponse ".($i+1)." est fausse";
                  $parfait = FALSE;
              }
          }
          if ($parfait)
              echo "<br />Bravo !";
          else
              echo "<br />Perdu ...";
      }
 ?>
</body>
</html>
```

Voici une capture d'écran de ce script en cours d'exécution.

Figure 3.7 : *Quizz v1*

Switch

Utiliser `switch` revient à utiliser des tests `if` consécutifs sur la même variable.

Voici un exemple d'utilisation de `switch`.

```
switch ($chanteur) {
    case "Cantat":
        echo "Ce chanteur est celui de Noir Desir";
        break;
    case "Harper":
        echo "Ce chanteur est celui de Ben Harper and the innocent criminals";
        break;
    case "Torrini":
        echo "Cette chanteuse est Emiliana Torrini";
        break;
    default:
        echo "Ce chanteur m'est inconnu";
}
```

Ce script est équivalent à :

```
if ($chanteur=="Cantat")
    echo "Ce chanteur est celui de Noir Desir";
elseif ($chanteur=="Harper")
    echo "Ce chanteur est celui de Ben Harper and the innocent criminals";
elseif ($chanteur=="Torrini")
    echo "Cette chanteuse est Emiliana Torrini";
else "Ce chanteur m'est inconnu";
```

`default` est la condition qui sera VRAIE si aucune des autres ne l'est.

L'instruction `break` est indispensable ici, sinon toutes les instructions qui suivent sont interprétées jusqu'au prochain `break` ou la fin du bloc `switch`. Cela peut être utile dans certains cas.

```
switch ($i) {
    case 0:
```

```
    case 1:
    case 2:
        echo "i est positif mais inférieur à 3 ";
        break;
    case 3:
        echo "i vaut 3";
}
```


REMARQUE

Comparaison
Contrairement à d'autres langages de programmation, la variable à comparer peut être aussi bien un nombre qu'une chaîne de caractères.

Break, continue

Deux instructions permettent de contrôler les sorties de boucle (`for`, `while`…). `break` est une instruction qui fait sortir de la boucle, alors que `continue` passe à l'itération suivante sans exécuter les instructions d'après.

```
<html>
    <head><title>Continue, Break</title></head>
    <body>
        <p>
<?php
    for ($i=0;$i<10;$i++) {
        echo $i;  // va afficher 0,2,4,6,8
        $i++;
        echo $i;  // va afficher 1.3.5.7.9
    }
    echo "<br>";
    for ($i = 0; $i < 10;$i++) {
        echo $i;
        $i++;
        if ($i>4) break;
        echo $i;
    }
    echo "<br>";
    for ($i=0;$i<10;$i++) {
        echo $i;
        $i++;
        if ($i>4) continue;
        echo $i;
    }
?>
        </p>
    </body>
</html>
```

Voici le résultat obtenu :

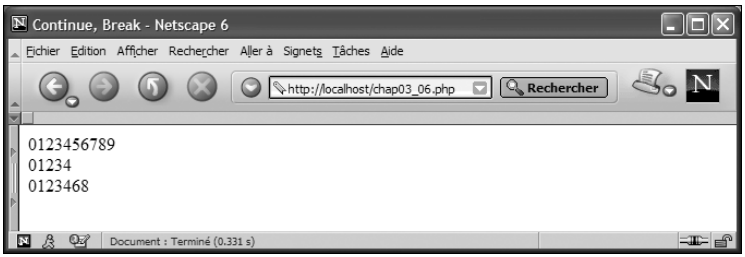


Figure 3.8 : *Résultat*

Pour bien comprendre, regardons la première boucle. Elle affiche tous les chiffres de 0 à 9, mais affiche deux chiffres à chaque itération.

La deuxième boucle, semblable à la première, utilise l'instruction `break`. À la première itération les chiffres 0 et 1 sont affichés. À la seconde les chiffres 2 et 3 sont affichés. Puis 4 est affiché. Enfin, après l'incrémentation `$i` vaut 5 (et la condition est vérifiée) et le programme sort définitivement de la boucle.

La troisième boucle utilise l'instruction `continue`. À la première itération les chiffres 0 et 1 sont affichés. À la seconde les chiffres 2 et 3 sont affichés. Puis 4 est affiché.Enfin, après l'incrémentation `$i` vaut 5 (la condition est vérifiée) et la seconde instruction `echo` n'est pas exécutée (donc 5 n'est pas affiché). Mais une nouvelle itération est réalisée faisant apparaître le chiffre 6, mais pas 7, puis 8, mais pas 9.

3.7. Les fonctions

PHP possède, comme de nombreux langages, la possibilité de regrouper des portions de code sous la forme de fonctions (ou procédures). Si la définition de fonctions n'a aucun impact sur le fonctionnement d'un programme, et donc s'il est possible de construire son développement comme une suite d'instructions mises bout à bout, elle procure en revanche visibilité et réduction de la taille du code source. La maintenance n'en est alors que plus simple et plus rapide.

Les fonctions permettant de structurer le code, il est fortement conseillé de les utiliser et de constituer des fichiers indépendants qui regrouperont les fonctions de même type. Ces bibliothèques pourront facilement être réutilisées dans d'autres développements et réduire ainsi le temps de mise au point (et donc le coût de production).

Trois avantages découlent donc de l'écriture du code sous forme de fonctions :

- Le temps de développement est considérablement réduit. En effet, vous n'avez pas à retaper des séquences de code identiques en divers endroits du programme. De la même façon, les bugs ne sont pas dupliqués, d'où un gain de temps considérable lors de la phase de mise au point.
- Meilleure lisibilité du code source. Le simple fait de regrouper le code en fonctions et les fonctions dans des fichiers séparés permet aux développeurs de s'y retrouver très facilement. De plus, donner à vos fonctions des noms significatifs est complémentaire avec l'ajout des commentaires, et facilite nécessairement la lisibilité et la compréhension du programme. Ceci est d'autant plus important pour la pérennité d'un code. Ainsi, la

maintenance et les modifications du programme sont facilitées, même pour un développeur reprenant son propre code quelques temps après l'avoir écrit.

- Le travail en équipe est amélioré. Un développement peut ainsi être découpé en plusieurs sous-parties, chaque développeur ayant alors la tâche de mettre au point une ou plusieurs fonctions.

Si l'emploi de fonctions est un avantage certain, une étape de conception préalable reste nécessaire afin de bien définir les fonctions qui devront être développées et de bien cibler leurs rôles respectifs. Plus important encore, l'étape préliminaire permet de fixer les règles d'entrée/sortie de chacune des fonctions, ceci afin que chaque développeur connaisse à l'avance les paramètres qu'il doit envoyer à la fonction et ce qu'elle doit retourner une fois le traitement effectué. Si les fonctions sont connues dès le début, alors chacun pourra commencer à les utiliser dans son code avant même la mise en production desdites fonctions.

La syntaxe

La fonction est un sous-programme isolé du reste du code et utilisable par un simple appel depuis n'importe quelle partie du programme.

La syntaxe pour déclarer une fonction simple est la suivante :

```
<?php
function premiereFonction()
{
    echo "J'aime PHP !";
}
?>
```

Ce code n'affiche rien tant que l'on ne fait pas appel à elle comme ceci :

```
<?php
function premiereFonction()
{
    echo "J'aime PHP !";
}
premiereFonction();  // Affiche "J'aime PHP !"
?>
```

Le cas précédent est un bon exemple de déclaration de fonction et d'utilisation de celle-ci à l'intérieur du programme principal. On voit l'intérêt qu'apporte la définition de la fonction dans le cas où la tâche de celle-ci est répétitive et devrait engendrer beaucoup de code source.

Le choix du nom d'une fonction

Afin de faciliter la lecture de votre code, n'hésitez pas à donner à vos fonctions un nom évocateur. Tout comme pour les variables, la lecture de votre programme est facilitée si ces éléments portent des noms explicites. Si les fonctions du programme s'appellent `fonction1()`, `fonction2()` *et* `fonction3()`, *la maintenance n'en sera que plus difficile. Ne donnez pas à vos fonctions un nom trop énigmatique ou qui peut avoir plusieurs sens. Évitez les noms de fonctions trop courts pour qu'ils ne*

soient pas utilisés plusieurs fois dans votre programme. Par exemple, pour une fonction devant compter un nombre de personnes présentes sur une page web, ne l'appelez pas simplement `compteur()`*, mais plutôt* `compteurPersonneEnLigne()`.

La portée des variables

Comme nous l'avons dit, une fonction n'est rien d'autre qu'un bout de code isolé du programme principal. C'est donc sans surprise que vous apprendrez qu'il est possible de définir des variables au sein des ses fonctions. Par contre, ce qu'il faut retenir c'est que les variables ainsi définies ne sont pas accessibles du reste du programme (elles ont une portée locale), comme le démontre l'exemple suivant :

```
<?php
function bicentenaire()
{
    $annee = "2002";
    echo "$annee c'était le bicentenaire de la naissance de Victor Hugo !";
}
bicentenaire();
// Affiche "2002 c'était le bicentenaire de la naissance de Victor Hugo !"
echo $année;  // n'affiche rien, la variable n'est pas définie.
?>
```

À l'inverse, une variable définie en dehors de la fonction (dans le programme principal) n'est pas accessible depuis la fonction.

```
<?php
$annee  = "2002";
$prenom = "Victor";
$nom    = "Hugo";

function bicentenaire()
{
    echo "$annee c'était le bicentenaire de la naissance de $prenom $nom !";
}
bicentenaire();
/*
affiche "c'était le bicentenaire de la naissance de !"
Les variables $annee, $nom et $prenom
ne sont pas définies au niveau de la fonction
*/
?>
```

Pour utiliser des variables globales dans la fonction, il faut utiliser le mot-clé `"global"` à l'intérieur de cette fonction, comme le montre l'exemple suivant :

```
<?php
$annee  = "2002";
```

```
$prenom = "Victor";
$nom    = "Hugo";

function bicentenaire()
{
    global $annee, $prenom, $nom;
    echo "$annee c'était le bicentenaire de la naissance de $prenom $nom !";
    $annee = "2003";
}

bicentenaire("bicentenaire");
  // affiche "2002 c'était le bicentenaire de la naissance de Victor Hugo!"

    echo $annee; // affiche 2003


?>
```

Comme vous le constatez, les modifications apportées à ces variables globales au sein de la fonction se répercutent en dehors de la fonction.

Il existe une autre façon de faire pour utiliser les variables globales depuis une fonction. Cette méthode consiste à utiliser directement le tableau prédéfini `$GLOBALS`. L'exemple précédent donnerait alors :

```
<?php
$annee  = "2002";
$prenom = "Victor";
$nom    = "Hugo";

function bicentenaire()
{
    echo $GLOBALS["annee"]." c'était le bicentenaire de la naissance de ".
         $GLOBALS["prenom"]." ".$GLOBALS["$nom"]." !";
    $GLOBALS["annee"] = "2003";
}

bicentenaire("bicentenaire");
 // affiche "2002 c'était le bicentenaire de la naissance de Victor Hugo!"

    echo $annee; // affiche 2003
?>
```

Variables d'environnement

Les variables externes `$_SERVER`, `$_ENV`, `$_GET`, `$_POST`, `$_SESSION`, `$_COOKIE`, `$_FILES`, etc. sont, quant à elles, accessibles depuis n'importe où. Il n'est donc pas nécessaire, dans leur cas, de préciser `global` pour les utiliser dans une fonction.

La portée d'une variable dépend donc de l'endroit où elle est initialisée. La portée d'une variable ne sera pas la même si elle est définie dans une fonction ou au début de votre programme.

On peut considérer trois niveaux de définition.

Tableau 3.12 : Les différents niveaux de définition des variables

Niveau	Description
Local	Les variables sont propres à chaque fonction. Dès la fin de l'exécution de cette fonction, la variable est détruite et l'espace mémoire qui contenait la valeur est libérée.
Global	Les variables globales sont définies pour la totalité du temps d'exécution du code de la page.
Static	Les variables statiques sont propres à chaque fonction, mais elles ne sont pas détruites à la fin de l'exécution de cette fonction. Si la fonction est rappelée plusieurs fois, la valeur de la variable est conservée et modifiée à chaque fois.

Il nous reste donc à aborder le cas des variables statiques. Ces variables sont initialisées au premier appel de la fonction et ne sont pas réinitialisées les fois suivantes. Pour créer une variable statique, il faut utiliser le mot-clé `static` suivi du nom de la variable.

Exemple d'utilisation :

```
<?
function maFonction (){
static $variable = "Emma ";

    $variable . =" C'est toi?";
    echo $variable."<br>";
}

maFonction();
maFonction();
maFonction();
?>
```

Cela affichera en effet :

```
Emma C'est toi?
Emma C'est toi? C'est toi?
Emma C'est toi? C'est toi? C'est toi?
```

Initialisation d'une variable statique
Il n'est pas possible d'initialiser une variable statique avec le résultat d'une fonction. On peut toutefois contourner ce problème en affectant à cette variable une valeur qu'elle est supposée ne jamais atteindre, et appeler la fonction si jamais la variable prend cette valeur.

```
<?php
function maFonction()
{
static $date = -1;
if ($date == -1) $date = time();
}
?>
```

Le passage des paramètres

Vous serez souvent amené à réclamer des paramètres pour vos fonctions. Pour cela, il suffit de compléter la ligne de déclaration de la fonction par une liste de noms de variables. Les variables ainsi définies ne sont accessibles qu'à l'intérieur de la fonction ; ce sont donc des variables de visibilité locale.

```
<?php
function bicentenaire($quoi)
{
    echo "2002 c'était le $quoi de Victor Hugo !";
}
bicentenaire("bicentenaire de la naissance ");
// Affiche "2002 c'était le bicentenaire de la naissance de Victor Hugo !"
echo $quoi;  // n'affiche rien, la variable n'est pas définie.
?>
```

Les paramètres par défaut

Vous pouvez donner à votre fonction des paramètres par défaut. Ainsi, si vous ne rentrez aucune valeur lors de l'appel de cette fonction dans votre code, une valeur par défaut est utilisée afin d'exécuter la fonction.

Pour placer une valeur par défaut, vous devez simplement spécifier cette valeur à l'aide du signe "=" lors de la déclaration de la fonction. Comme ceci :

```
<?php
function exemple($parDefaut="bonjour")
{
    return $parDefaut;
}
?>
```

Cette fonction peut donc recevoir ou non une valeur comme paramètre. Si ce n'est pas le cas, alors celle-ci prend, par défaut, la valeur `"bonjour"`. Il y a donc deux façons d'appeler cette fonction dans le programme principal.

```
<?php
function exemple($pardefaut="bonjour")
{
    return $pardefaut;
}
```

```
// soit en plaçant une valeur en paramètre
echo exemple("bye bye");  // affiche "bye bye"
// soit en laissant vide
echo exemple();  // affiche "bonjour"
?>
```

Le fait que les fonctions nécessitent a priori un nombre fixe d'arguments vous posera peut-être un jour un problème. Mais il y a deux façons d'y faire face:

- En utilisant un tableau, si votre fonction n'accepte qu'un seul argument mais que celui-ci est de type tableau, vous pouvez y mettre autant d'arguments que vous voulez.
- En utilisant les fonctions mises à disposition par PHP (ce qui en fait s'apparente à utiliser un tableau créé par PHP).

Ainsi, alors que `func_num_args()` renvoie le nombre d'éléments passés en paramètre d'une fonction, `func_get_arg()` et `func_get_args()` permettent de récupérer un argument ou un tableau d'arguments.

func_num_args()

Retourne le nombre d'arguments passés à une fonction.

Syntaxe	`int func_num_args(void)`
`retour`	Retourne un nombre correspondant aux nombres d'éléments passés comme arguments de la fonction.

```
<?php
function elementsVariables()
{
    return func_num_args();
}

echo elementsVariables();  // Affiche 0;
echo elementsVariables(1, 2, 3);  // Affiche 3;
echo elementsVariables("toto", 5, TRUE);  // Affiche 3;
?>
```

func_get_arg()

Retourne la valeur d'un des éléments passés en argument de la fonction.

Syntaxe	`mixed func_get_arg(int $numArg)`
`$numArg`	Position de l'élément à consulter.
`retour`	La valeur de l'argument.

func_get_args()

Retourne un tableau contenant tous les éléments passés en argument à la fonction.

Syntaxe	`array func_get_args(void)`
`retour`	Tableau indexé contenant la liste des arguments de la fonction.

Voici un exemple de l'utilisation qui peut être faite des trois fonctions précédemment décrites :

```
<?php
function elementsVariables()
{
    // Compte le nombre d'éléments
    $nombreElements = func_num_args();
    // Récupère tous les éléments dans un tableau
    $tableauElements = func_get_args();

    for ($i=0; $i<$nombreElements; $i++)
    {
        // Affiche tous les arguments de la fonction
        echo "élément $i : ".$tableauElements[$i]."<br />";
    }

    // Récupère l'élément désigné par le deuxième argument
    echo "l'argument n°".func_get_arg(1).
    " = ".func_get_arg(func_get_arg(1))."<br />";
}

elementsVariables(10, 0, 3);

echo "<br />";

elementsVariables("toto", 3, TRUE, "PHP");
?>
```

L'exécution de ce programme renvoie donc ici :

```
élément 0 : 10
élément 1 : 0
élément 2 : 3
l'argument n°0 = 10

élément 0 : toto
élément 1 : 3
élément 2 : 1
élément 3 : PHP
l'argument n°3 = PHP
```

Le passage de paramètres par référence

Lorsque l'on passe une variable en paramètre à une fonction, la variable transmise n'a qu'une portée locale, et la variable manipulée au sein de la fonction n'est en fait qu'une copie de la variable transmise par le programme appelant. Ainsi, l'exécution de la fonction n'entraîne aucune modification sur la variable transmise.

Lorsque l'on souhaite que la fonction modifie la valeur de la variable passée en paramètre, celle-ci doit être passée non pas par valeur mais par référence. Lorsqu'une variable est passée en référence, la nouvelle variable locale ainsi créée pointe sur la même zone mémoire, mais avec un autre nom que le paramètre. Ainsi, on a un alias de la variable qui se crée dans la fonction, et toutes les modifications qui lui seront apportées seront répercutées sur l'original.

Ce n'est toutefois pas l'appelant qui décide si la variable doit être passée par valeur ou par référence, mais la fonction. Ainsi, pour utiliser une référence, il suffit simplement de signaler le paramètre de la fonction avec un "`&`" devant le nom de la variable.

```
<?
$phrase = "Ceci est temporaire";

function citation(&$reference, $nom, $prenom)
{
    $reference = $nom." ".$prenom.
                " a écrit : Les écrivains ont mis la langue en liberté."
}

citation($phrase, "Victor", "Hugo");
echo $phrase;
// affiche "Victor Hugo a écrit : Les écrivains ont mis la langue en liberté."
?>
```

Dans l'exemple précédent, nous pouvons constater que les modifications apportées à la variable locale `$reference` se répercutent sur la variable globale `$phrase`.

Utilisation des paramètres de type objet

Comme nous le verrons dans le chapitre sur la programmation orientée objet, contrairement au comportement d'autres langages de programmation, dans les versions de PHP inférieure à la version 5 (incluant le moteur Zend2), les objets ne sont pas automatiquement passés par référence. Pour ne pas travailler sur une copie de l'objet il est donc, là aussi, nécessaire d'utiliser le "et commercial" '&'.

PHP 5

Avec PHP 5, il est désormais possible de spécifier une valeur par défaut là où l'on attend un argument passé par référence.

```
function maFonction(&$param = "valeurParDefaut")
```

Alors que, vous l'aurez deviné, ce n'était pas possible auparavant.

Retourner une valeur

Vous serez sans doute amené à vouloir récupérer le contenu d'une variable à la suite du traitement d'une fonction. Pour effectuer le renvoi d'une variable et ainsi terminer l'exécution de la fonction, le langage PHP utilise le mot-clé `return`. Lorsque cette instruction est rencontrée, la fonction évalue la valeur de la variable qui suit et la renvoie dans le programme principal.

Exemple d'utilisation :

```
return $valRetour;
return 0;
return "Laurent";
```

Ce qui donne dans le cas d'une utilisation :

```
<?php
function fonctionRetour()
{
    return "guitare";
}
echo fonctionRetour();  // affiche "guitare"
?>
```

S'il est possible de placer plusieurs instructions `return` à l'intérieur d'une fonction, le premier `return` exécuté met un terme à l'exécution de la fonction.

```
<?php
function associationNom($varATester)
{
    if ($varATester=="Laurent") {
        return "GUEDON";
    } elseif ($varATester =="Damien" || $varATester == "Thomas") {
        return "HEUTE";
    } elseif ($varATester =="Pierre-Emmanuel") {
        return "MULLER";
    } else {
        return "Qui ???";
    }
}
?>
```

L'instruction `return` ne permet de renvoyer qu'une seule valeur. Si vous souhaitez retourner plusieurs valeurs, deux possibilités s'offrent à vous :

- Retourner un tableau de valeurs ou un objet avec plusieurs attributs.
- Utiliser plusieurs paramètres passés par référence (qui serviront donc plus de valeur retour que de valeur d'entrée) ; éventuellement, la fonction pourra également retourner une valeur via `return`.

Il n'est pas vraiment possible de dire a priori quelle est la bonne méthode. C'est un peu au cas par cas. La première oblige ensuite à analyser le contenu d'un tableau (et l'utilisation d'un objet

ne se justifie pas toujours). La seconde méthode n'est pas totalement satisfaisante, puisque l'on mélange paramètres d'entrée et paramètres de sortie ; a contrario, elle permet de réserver la valeur retour pour un booléen indiquant si l'opération s'est bien passée ou non.

Voici un exemple utilisant un tableau montrant comment sortir de ce problème, que vous rencontrerez sans doute dans votre carrière de développeur.

```
<?php
function discotheque($choix)
{
    switch ($choix) {
        case 1 :
            $artiste = "Placebo";
            $titre = "Where is my mind";
            break;
        case 2 :
            $artiste = "Emiliana Torrini";
            $titre = "To be free";
        defaut :
            $artiste = "Radiohead";
            $titre = "Creep";
    }
    return array($artiste, $titre);
}

$disque = discotheque(3);

echo "Artiste ".$disque[0] ."\n";
echo "Titre ".$disque[1] ."\n";
?>
```

Ici, la fonction retourne un tableau de valeurs. Il est donc ensuite possible de traiter le tableau et de sortir les différentes données, comme si la fonction nous avait renvoyé plusieurs variables. Le résultat du petit programme ci-dessus nous donne donc :

```
Artiste Radiohead
Titre Creep
```

Manipuler des fonctions

Une série de commandes permet de créer à la volée, de tester l'existence des fonctions et de les manipuler. L'une d'elle permet de créer des fonctions dites anonymes.

create_function()

Permet de créer une fonction anonyme à partir de paramètres.

Syntaxe	`string create_function(string $arguments ,string $code )`
`$arguments`	Liste des paramètres de la fonction séparés par une virgule.
`$code`	Le code de la fonction.
`retour`	Nom attribué à la fonction (en s'assurant de son unicité).

```
<?php
$fonction = create_function('$a,$b','return $a + $b;');
echo "Nom de la nouvelle fonction : $fonction<br />";
echo "10 + 5 = ".$fonction(10, 5);
// "Nom de la nouvelle fonction : lambda_1"
// "10 + 5 = 15"
?>
```

retourne

```
Nom de la nouvelle fonction: lambda_1
10 + 5 = 15
```

Notez que vous pouvez utiliser des guillemets (à la place des apostrophes) dans la définition des arguments et du code de la fonction, à condition de ne pas oublier d'échapper le caractère `$` précédant les noms de variables comme ceci :

```
$fonction = create_function("\$a,\$b","return \$a + \$b;");
```

Il est également possible de vérifier si une fonction existe ou non. Ceci permet notamment de vérifier si l'environnement (version de PHP, options de configuration) dans lequel tourne le script permet la réalisation de l'opération prévue. Ceci peut également permettre de créer une fonction émulant une fonction existant dans les dernières versions de PHP, uniquement si celle-ci n'est pas disponible dans la version utilisée (afin que le script puisse fonctionner sur d'anciennes versions).

function_exists()

Permet de vérifier si une fonction existe bien.

Syntaxe	`boolean function_exists(function $nomFonction)`
`$nomFonction`	Nom de la fonction à vérifier.
`retour`	Retourne `TRUE` si la fonction a été trouvée, `FALSE` sinon.

```
<?php
if (function_exists("uneFonctionDontJAiBesoin")) {
    echo "Chouette je peux utiliser la fonction<br />";
    uneFonctionDontJAiBesoin();
} else {
    echo "La fonction n'existe pas tant pis pour vous<br />";
}
```

```
if (function_exists("function_exists")) {
    echo "Par contre la fonction function_exists() existe<br />";
} else {
    echo "Curieux<br />";
}
```

retournera

```
La fonction n'existe pas tant pis pour vous
Par contre la fonction function_exists() existe
```

On peut aussi simplement lister toutes les fonctions définies à l'aide de la fonction `get_defined_functions()`.

get_defined_functions()

Liste toutes les fonctions définies.

Syntaxe	`array get_defined_functions(void)`
`retour`	Tableau associatif contenant les clés. "`internal`" ayant pour valeur un tableau indexé contenant les noms des fonctions PHP "natives". "`user`" ayant pour valeur un tableau indexé contenant les noms des fonctions définies par l'utilisateur.

```
<?php
function fonction1($a, $b)
{
    return ($a + $b);
}

function fonction2($a)
{
    return $a;
}

function fonction3($a, $b)
{
    return ($a + $b);
}

function fonction4($a, $b, $c)
{
    return ($a + $b + $c);
}

function fonction5($a, $b, $c, $d)
{
```

```
    return ($a + $b + $c + $d);
}

$tableauFonction = get_defined_functions();
print_r($tableauFonction);
?>
```

Cela affiche toutes les fonctions sous la forme d'un tableau de tableaux.

```
Array
(
    [internal] => Array
        (
            [0] => zend_version
            [1] => func_num_args
            [2] => func_get_arg
            ......
            [1161] => apache_note
            [1162] => apache_lookup_uri
            [1163] => apache_child_terminate
        )
    [user] => Array
        (
            [0] => fonction1
            [1] => fonction2
            [2] => fonction3
            [3] => fonction4
            [4] => fonction5
        )
)
```

Listons simplement les fonctions utilisateurs.

```
<?php
while(list ($key, $val)=each($tableauFonction["user"]))
{
    echo "$key => $val<br />";
}
?>
```

Le résultat cette fois est simplement :

```
0 => fonction1
1 => fonction2
2 => fonction3
3 => fonction4
4 => fonction5
```

Les fonctions utilisateurs peuvent aussi être appelées à l'aide des commandes `call_user_func_array()` ou `call_user_func()`. Ces commandes sont très utiles lorsqu'un programme doit faire appel à différentes fonctions dynamiquement.

call_user_func()

Appelle une fonction utilisateur en lui transmettant des paramètres.

Syntaxe	`mixed call_user_func(string $nomFonction [, mixed $parametre [, mixed ...]])`
`$nomFonction`	Nom de la fonction à appeler.
`$parametre`	Liste des paramètres (à transmettre dans le même ordre que sa déclaration).
`retour`	Renvoie le code retour de la fonction appelée.

call_user_func_array()

Appelle une fonction utilisateur en lui transmettant des paramètres rassemblés sous la forme d'un tableau.

Syntaxe	`mixed call_user_func_array(string $nomFonction [, array $parametre])`
`$nomFonction`	Nom de la fonction à appeler.
`$parametre`	Liste des paramètres (à transmettre sous la forme d'un tableau).
`retour`	Renvoie le code retour de la fonction appelée.

```
<?php
function nombreCaractere($phrase) {
    return strlen ($phrase);
}

echo "phrase 1 = ".call_user_func('nombreCaractere',
'Ich bin ein Berliner! - JF Kennedy(1963)')." caractères<br />";
echo "phrase 2 = ".call_user_func('nombreCaractere',
'13 janvier 1898, Zola publia J'accuse')." caractères<br />";
?>
```

Le résultat de ce programme affiche ici :

```
phrase 1 = 40 caractères
phrase 2 = 38 caractères
```

```
<?php
function fonction1($a, $b) { return $a+$b; }
function fonction2($a, $b) { return $a-$b; }
function fonction3($a, $b) { return $a*$b; }
function fonction4($a, $b) { return $a/$b; }

$tableau = array(10,5);
```

```
for ($i=1;$i<=4;$i++)
{
    echo "$i => ".call_user_func_array('fonction'.$i, $tableau)."<br />";
}
?>
```

```
1 => 15
2 => 5
3 => 50
4 => 2
```


La récursivité des fonctions

Une fonction est dite récursive si elle s'appelle elle-même. Cette utilisation particulière des fonctions est rare (ce qui, entre nous, n'est pas forcément un défaut, car elle est très gourmande en mémoire.). Et pourtant, si elle n'est pas indispensable, elle peut-être très pratique lorsqu'il faut traiter certains problèmes comme le parcours d'une arborescence.

Un exemple de fonction récursive est celui de la fonction donnant le factoriel d'un nombre.

```
<?php
function factorielle($nombre)
{
    if ($nombre==0)
    {
        return "1"."<br />";
    }else{
        return $nombre*factorielle($nombre-1)."<br />";
    }
}
echo factorielle (3);  // 3! = 1x2x3 = 6
echo factorielle (3);  // 3! = 1x2x3 = 6
?>
```

Juste un exemple... de ce qu'il ne faut pas faire

Si cette méthode est la plus simple pour montrer ce qu'est une fonction factorielle, c'est aussi la plus mauvaise méthode pour implémenter factoriel.

De plus, les fonctions récursives sont très délicates à manipuler. Il est très important de vérifier que votre fonction se termine bien à un moment donné pour éviter des problèmes de boucles infinies.

Un des exemples de fonctions récursives les plus répandus est la résolution du problème de la tour de Hanoï. La tour de Hanoï se présente comme un casse-tête. Plusieurs disques sont empilés sur une tige, du plus large au plus petit. Deux autres tiges se trouvent à côté et le but est de déplacer les disques pour les superposer sur la tige centrale dans le même ordre, c'est-à-dire du plus grand au plus petit.

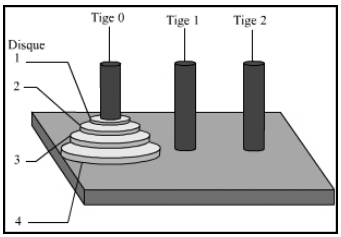


Figure 3.9 :
La tour de Hanoï

```
<html>
<body>
<?php
function deplaceDisque($disque, $tige1, $tige2)
{
    echo "Déplacer le disque ".$disque." sur la tige ".$tige2."<br />";
}

function hanoi($disque, $tige1, $tige2)
{
    if ($disque == 1) // si le disque à déplacer est le premier
    {
        // on ne fait qu'un déplacement
        deplaceDisque (1, $tige1, $tige2);
    }else{
        // sinon il faut déplacer $disque-1
        // sur la tige différente de $tige1 et $tige2
        hanoi($disque-1, $tige1, 3-$tige1-$tige2);
        deplaceDisque ($disque, $tige1, $tige2);
        // et on redéplace les ($disque-1) disques sur le disque déplacé
        hanoi($disque-1, 3-$tige1-$tige2, $tige2);
    }
}

hanoi(4, 0, 1);  // 4 disques utilisés
?>
</body>
</html>
```

3.8. Les tableaux

En PHP, il est possible de distinguer trois types de tableaux.

- Les tableaux indexés où chaque élément du tableau porte un numéro (ce qui correspond à la représentation habituelle d'un tableau dans les autres langages de programmation). Par défaut, la numérotation commence avec l'indice 0.
- Les tableaux associatifs où chaque élément du tableau est associé à une clé représentée par une chaîne de caractères (ce qui correspond à peu de choses près à une table de hash dans les autres langages de programmation).

- Les tableaux mixtes où chaque élément du tableau est associé aussi bien à un indice qu'à une clé.

Cette distinction entre les tableaux est surtout "conceptuelle". En effet, leur mode de fonctionnement est totalement identique.

Index et ordre dans le tableau
Les index doivent être perçus simplement comme des clés de type entier. La valeur d'un index ne préjuge pas de la position d'un élément dans le tableau. Un élément peut avoir un index égal à 0 et se trouver en fin de tableau.
Un tableau est juste un ensemble (ordonné) de couple (clé, valeur).

Les valeurs d'un tableau

Les valeurs contenues dans un tableau peuvent être de tout type connu. Ainsi, un tableau peut stocker des entiers, des réels, des chaînes de caractères, des tableaux, des objets, etc.

De plus, les types des valeurs contenues dans un tableau peuvent varier d'un indice (ou d'une clé) à l'autre. Le second élément d'un tableau pourra contenir une chaîne de caractères, même si le premier élément contient un objet.

Initialisation d'un tableau

La syntaxe de base de l'initialisation d'un tableau est la suivante :

```
$monTableau = array();
```

Mais, à part pour forcer le type d'une variable au type tableau, il est bien rare que nous utilisions cette forme épurée.

L'initialisation du tableau se fait généralement en même temps que son remplissage.

Ainsi, dans le cas "classique" d'un tableau indexé, on pourra par exemple préciser les valeurs associées aux trois premiers index par :

```
$monTableau = array(23, "PHP", 12.4);
```

Ce qui donnera alors :

```
$monTableau[0] = 23;
$monTableau[1] = "PHP";
$monTableau[2] = "12.4";
```

Dans le cas d'un tableau associatif, il faudra préciser la clé et la valeur selon le schéma suivant :

```
$tableauAge = array("Pierre" => 22, "Thierry" => 29, "Jean" => 34);
```

Pour obtenir :

```
$tableauAge["Pierre"] = 22;
$tableauAge["Thierry"] = 29;
$tableauAge["Jean"] = 34;
```

Et pour un tableau mixte il est possible de préciser :

```
$monTableau = array("Thierry", 29, "Prénom" => "Thierry", "Age" => 29);
```

Ce qui donnerait :

```
$monTableau[0] = "Thierry";
$monTableau[1] = 29;
$monTableau["Prénom"] = "Thierry";
$monTableau["Age"] = 29;
```


Cas d'utilisation des tableaux mixtes
Les tableaux mixtes sont utilisés en particulier pour la lecture des enregistrements d'une base de données. Ainsi, l'utilisateur peut faire appel à la valeur d'un champ soit à partir de son index soit à partir du nom du champ.

Les subtilités d'initialisation d'un tableau

Même dans le cas d'un tableau indexé, il est possible de préciser pour quel indice vous êtes en train de définir la valeur (à la manière des tableaux associatifs).

```
$monTableau = array(1 => "Valeur1", 3 => "Valeur3");
```

donnera les valeurs suivantes :

```
$monTableau[1] = "Valeur1";
$monTableau[3] = "Valeur3";
```

`$monTableau[0]` et `$monTableau[2]`, quant à eux, n'auront pas de valeur affectée.

Ordre d'affectation
Il n'est pas nécessaire de préciser les valeurs dans l'ordre des indices. Ainsi

```
$monTableau = array(3 => "Valeur3", 1 => "Valeur1");
```

est tout à fait valide.

Là encore, il est possible, lors de l'initialisation d'un tableau, de mélanger cette notation avec celle d'un tableau associatif, et même avec la notation "traditionnelle" d'un tableau indexé. Dans ce dernier cas, les valeurs pour lesquelles on ne précise pas les index auront nécessairement pour index le dernier index affecté + 1 (ou par défaut 0).

```
$monTableau = array("Valeur0", 2 => "Valeur2", "Valeur3");
```

donnera :

```
$monTableau[0] = "Valeur0";
$monTableau[2] = "Valeur2";
$monTableau[3] = "Valeur3";
```

Commencer un tableau à l'indice 1
Cette propriété est bien pratique pour commencer un tableau avec l'indice 1.

```
$monTableau(1 => "Valeur1", "Valeur2", "Valeur3");
```

Dans le cas d'un tableau indexé ou associatif, si, dans la chaîne d'initialisation, plusieurs valeurs sont affectées à un même indice ou à une même clé, la valeur conservée sera toujours la dernière rencontrée.

Ainsi,

```
$monTableau = array("Valeur qui va se faire écraser",
                    0 => "Oups... j'écrase");
```

donnera :

```
$monTableau[0] = "Oups... j'écrase";
```

Remplissage d'un tableau

D'une manière tout à fait classique, le remplissage (où l'affectation des valeurs) d'un tableau se fait par un appel du type :

```
$monTableau[0] = "Valeur";
```

ou

```
$monTableau["cle"] = "Valeur";
```

Mais une des particularités du langage PHP est que la taille des tableaux n'est pas fixe. Elle peut donc être augmentée au fil des besoins. Ainsi, pour un tableau indexé qui a été initialisé avec trois éléments, il est possible d'affecter une valeur à l'élément d'indice 10, comme le montre l'exemple suivant :

```
<?php
$monTableau = array("valeur1, "valeur2", "valeur3");
$monTableau[10] = "valeur10",
?>
```

De même, pour un tableau associatif, il est toujours possible d'ajouter un élément avec une nouvelle clé.

Cela est particulièrement utile pour composer une liste d'éléments. De plus, pour simplifier la tâche, le langage PHP nous permet de ne pas préciser d'indice ou de clé lors d'une affectation, ceci afin d'utiliser pour indice le dernier indice affecté + 1.

```
$monTableau[] = "Valeur d'indice 0";
$monTableau[] = "Valeur d'indice 1";
```

sera alors équivalent à :

```
$monTableau[0] = "Valeur d'indice 0";
$monTableau[1] = "Valeur d'indice 1";
```

Les fonctions de manipulation des tableaux

Il existe de nombreuses fonctions liées au traitement des tableaux. Certaines sont destinées au parcours du tableau, d'autres à la fusion de tableaux, d'autres encore au tri des valeurs ou des clés, etc.

Vous découvrirez également dans le chapitre dédié à la programmation orientée objet "Les classes, les objets" qu'il existe également des objets permettant la manipulation de tableaux.

Fonctions de base

is_array()

Indique si une variable est de type tableau.

Syntaxe	`boolean is_array(mixed $variable)`
`$variable`	Variable à tester.
`retour`	`TRUE` si la variable est de type tableau, `FALSE` sinon.

count()

Retourne le nombre d'éléments contenus dans un tableau.

Syntaxe	`int count(array $tableau)`
`$tableau`	Le tableau dont on veut compter le nombre d'éléments.
`retour`	Le nombre d'éléments du tableau. Si toutefois le paramètre fourni ne correspond pas à un tableau mais à une variable existante, alors la valeur retournée est `1`. S'il s'agit d'une variable qui n'existe pas, la fonction retourne `0`.

S'il s'agit d'un tableau indexé avec des valeurs pour chaque index à partir de 0, alors on pourra afficher son contenu grâce au code suivant :

```
<?php
    $monTableau = array ("PHP", "C'est", "vraiment", "sympa");
    for ($i=0; $i<count($monTableau); $i++) echo $monTableau[$i]."<br />";
?>
```

La fonction print_r()
La fonction `print_r()` est également une instruction fort pratique pour visualiser le contenu d'un tableau (principalement à titre de débogage).

sizeof()

Est l'équivalent de `count()` (même syntaxe, même comportement).

array_values()

Retourne un tableau indexé contenant les valeurs des éléments d'un tableau donné (ce qui peut permettre de réindexer un tableau ou de convertir un tableau associatif en tableau indexé).

Syntaxe	`array array_values(array $tableau)`
`$tableau`	Tableau de référence.
`retour`	Tableau indexé des valeurs trouvées dans `$tableau`.

Voir aussi `array_keys()`, qui peut être utilisé pour faire des recherches dans un tableau.

array_unique()

Retourne un tableau dans lequel aucune valeur n'apparaît plusieurs fois (associées à différentes clés).

Syntaxe	`array array_unique(array $tableau)`
`$tableau`	Tableau de référence.
`retour`	Tableau dans lequel chaque valeur n'apparaît qu'une seule fois. C'est toujours la dernière clé associée à une valeur donnée qui est conservée.

array_flip()

Intervertit les rôles des clés (ou index) et des valeurs d'un tableau. Cette fonction n'est applicable que si les valeurs peuvent devenir des clés ou, autrement dit, si les valeurs sont des chaînes de caractères ou des entiers.

Si le tableau possède plusieurs valeurs identiques, seule la dernière paire (clé, valeur) sera considérée.

Syntaxe	`array array_flip(array $tableau)`
`$tableau`	Tableau pour lequel clés et valeurs doivent être permutées.
`retour`	Tableau pour lequel les clés et les valeurs ont été permutées, ou `FALSE` en cas d'échec.

array_rand()

Retourne un index (ou une clé) ou un tableau d'index (ou de clés) distincts pris pseudo-aléatoirement dans un tableau.

Syntaxe	`mixed array_rand(array $tableau, [int $nbCles])`
`$tableau`	Tableau sur lequel s'effectue la recherche.
`$nbCles`	Nombre de clés ou index à retourner (par défaut le nombre de clés est fixé à 1).
`retour`	Un index (ou une clé) ou un tableau d'index (ou de clés) distincts pris pseudo-aléatoirement dans le tableau.

Pseudo-aléatoire
Comme pour toutes les fonctions pseudo-aléatoires, il est fortement conseillé de faire au préalable un appel à la fonction `srand()` pour "taper" un peu n'importe où dans la pile des nombres pseudo-aléatoires.

Vous pouvez vous reporter au chapitre "Les fonctions mathématiques" pour plus de détails sur les nombres pseudo-aléatoires et la fonction `srand()`.

Fonctions de recherche dans les tableaux

array_keys()

Retourne un tableau indexé contenant les clés et index utilisés dans un tableau donné ou, éventuellement, les clés et index associés à une valeur donnée.

Syntaxe	`array array_keys(array $tableau [, mixed $valeur])`
`$tableau`	Tableau de référence.

`$valeur`	Argument optionnel précisant à quelle valeur doivent être associés les clés et index retournés. Par défaut, ce sont tous les index et clés du tableau qui sont retournés.
`retour`	Tableau des clés et index trouvés ou un tableau vide si la valeur n'est pas trouvée.

Listing 3.11 : array_array_keys.php

```
<?php
    $sportifs = array("Z. Zidane" => "Foot", "C. Moreau" => "Cyclisme",
                      "T. Henry" => "Foot", "M. Indurain" => "Cyclisme",
                      "J. Durand" => "Cyclisme", "T. Marie" => "Cyclisme");

    $cyclistes = array_keys($sportifs, "Cyclisme");

    echo "Voici une liste de cyclistes <br />";
    for ($i=0; $i<count($cyclistes); $i++) {
        echo $cyclistes[$i]."<br />";
    }
?>
```

retournera bien les noms associés au cyclisme.

```
Voici une liste de cyclistes
C. Moreau
M. Indurain
J. Durand
T. Marie
```

array_search()

Retourne la première clé (ou index) du tableau, associée à une valeur donnée.

Syntaxe	`mixed array_search(mixed $valeur, array $tableau [,boolean $strict])`
`$valeur`	Valeur recherchée.
`$tableau`	Tableau sur lequel se porte la recherche.
`$strict`	Argument optionnel à positionner à `TRUE` si l'on veut que la comparaison tienne également compte du type de la valeur cherchée. `FALSE`, par défaut.
`retour`	Clé (ou index) trouvée, `FALSE` sinon.

array_key_exists()

Indique si un tableau contient ou non une clé donnée.

Syntaxe	`boolean array_key_exists(mixed $cle, array $tableau)`
`$cle`	Clé recherchée.
`$tableau`	Tableau sur lequel se porte la recherche.
`retour`	`TRUE` si l'index existe, `FALSE` sinon.

in_array()

Indique si un tableau contient ou non une valeur donnée.

Syntaxe	`boolean in_array(mixed $valeur, array $tableau [, boolean $strict])`
`$valeur`	Valeur recherchée.
`$tableau`	Tableau sur lequel se porte la recherche.
`$strict`	Argument optionnel à positionner à `TRUE` si l'on veut que la comparaison tienne également compte du type de la valeur cherchée. `FALSE`, par défaut.
`retour`	`TRUE` si la valeur a été trouvée dans `$tableau`. `FALSE` sinon.

Fonctions de manipulation de portions de tableau

range()

Retourne un tableau composé des entiers compris entre une valeur de début et une valeur de fin.

Syntaxe	`array range(int $debut, int $fin)`
`$debut`	Premier entier du tableau.
`$fin`	Dernier entier du tableau (doit nécessairement être supérieur ou égal à `$debut`).
`retour`	Tableau composé des entiers compris entre la valeur de début (incluse) et la valeur de fin (inclus) ou bien un tableau vide en cas d'échec.

array_fill()

Retourne un tableau indexé construit à partir de la répétition d'une valeur.

Syntaxe	`array array_fill(int $indexDebut, int $nombre, mixed $valeur)`
`$indexDebut`	Premier index du tableau.
`$nombre`	Nombre d'éléments dans le tableau.
`$valeur`	Valeur à donner à chaque élément du tableau.
`retour`	Tableau indexé de `$indexDebut` à `$indexDebut+$nombre-1` ne contenant que la valeur `$valeur`.

array_pad()

Retourne un tableau complété (à droite ou à gauche) avec une valeur donnée pour atteindre un nombre total d'éléments spécifiés.

Syntaxe	`array array_pad(array $tableau, int $taille, mixed $valeur)`
`$tableau`	Tableau de référence.
`$taille`	Taille (en valeur absolue) souhaitée pour le tableau résultat. Si vous souhaitez compléter le tableau à gauche, indiquez alors la valeur en négatif.
`$valeur`	Valeur à utiliser pour compléter le tableau.
`retour`	Retourne le tableau spécifié, complété par la valeur pour atteindre la taille demandée. Si celle-ci (en valeur absolue) est inférieure à la taille initiale du tableau, ce dernier n'est tout simplement pas complété, mais sa taille n'en est pas pour autant réduite.

array_slice()

Retourne un tableau extrait d'un autre tableau.

Syntaxe	`array array_slice(array $tableau, int $debut [, int $longueur])`
`$tableau`	Tableau de référence.
`$debut`	Numéro d'ordre du premier élément du tableau à extraire. Au premier élément du tableau correspond le numéro 0. Ce numéro d'ordre est indépendant de l'index (en particulier après un appel à la fonction `array_reverse()` avec comme second argument la valeur `TRUE`, l'élément d'index 0 peut se retrouver en fin de tableau). Si `$debut` est négatif, alors le compte se fait en partant de la fin du tableau.

`$longueur`	Argument optionnel précisant le nombre d'éléments à extraire. Par défaut, ce sont tous les éléments suivants du tableau qui sont extraits. Si `$longueur` est négatif, ce sont tous les éléments jusqu'à l'élément qui se trouve à `abs($longueur)` de la fin qui sont extraits.
`retour`	Tableau des éléments extraits. Les clés sont conservées mais pas les index.

Listing 3.12 : array_array_slice.php

```
<?php
    $tableau = array("cle" => "element0", 2 => "element1",
                     1 => "element2");

    print_r(array_slice($tableau, 0, 2));
?>
```

retourne par exemple :

Array ([cle] => element0 [0] => element1)

array_splice()

Supprime les éléments d'un tableau ; éventuellement, insère des éléments dans le tableau et retourne le tableau des éléments supprimés.

Syntaxe	`array array_splice(array $tableau, int $debut [, int $longueur [, array $remplacement]])`
`$tableau`	Tableau à modifier.
`$debut`	Numéro d'ordre du premier élément du tableau à supprimer. Au premier élément du tableau correspond le numéro 0. Ce numéro d'ordre est indépendant de l'index (en particulier après un appel à la fonction `array_reverse()` avec comme second argument la valeur `TRUE`, l'élément d'index 0 peut se retrouver en fin de tableau). Si `$debut` est négatif, alors le compte se fait en partant de la fin du tableau.
`$longueur`	Argument optionnel précisant le nombre d'éléments à supprimer. Par défaut, ce sont tous les éléments suivants du tableau qui sont supprimés. Si `$longueur` est négatif, ce sont tous les éléments jusqu'à l'élément qui se trouve à `abs($longueur)` de la fin qui sont supprimés.
`$remplacement`	Tableau optionnel contenant les valeurs à insérer dans le tableau à partir de l'élément pointé par `$début`. Si le nombre d'éléments insérés est supérieur au nombre d'éléments supprimés, alors le reste du tableau est décalé. Dans ce cas, les clés sont conservées mais pas les index. Par défaut, les éléments sont simplement supprimés.
`retour`	Tableau des éléments supprimés.

array_chunk()

Retourne un tableau indexé ayant pour valeurs les morceaux d'un tableau découpé en morceaux.

Syntaxe	`array array_chunk(array $tableau, int $taille [, boolean $mode])`
`$tableau`	Tableau à découper en morceaux.
`$taille`	Taille des morceaux (nombre d'éléments).
`$mode`	Précise si les clés du tableau doivent être conservées : `TRUE`, les clés sont conservées. `FALSE` (valeur par défaut), chaque morceau de tableau est réindexé en commençant par 0.
`retour`	Tableau indexé contenant des tableaux issus de la découpe en morceaux du tableau d'entrée.

Fonctions de manipulation des clés d'un tableau

array_change_key_case()

Retourne un tableau avec la casse des clés modifiée.

Syntaxe	`array array_change_key_case(array $tableau [, int $casse])`
`$tableau`	Tableau de référence.
`$casse`	Au choix, l'une des deux constantes suivantes : `CASE_LOWER` (valeur par défaut), pour mettre toutes les clés en minuscules. `CASE_UPPER`, pour mettre toutes les clés en majuscules.
`retour`	Tableau avec tous les index en majuscules ou minuscules.

Fonctions de conversion tableau <-> variable

list()

Construit une liste de variables à partir des données d'un tableau.

Ce n'est pas une véritable fonction, puisqu'elle s'utilise de la façon suivante :

```
list($variable1, $variable2, ...) = $tableau
```

Syntaxe	`void list(mixed $variable1 [,mixed $variable2, ...])`
`$variable1`	Variable à laquelle doit être affectée la première valeur du tableau.
`$variable2`	Variable à laquelle doit être affectée la seconde valeur du tableau.
`...`	etc.

extract()

Construit une liste de variables à partir des données d'un tableau, les noms des variables étant basés sur les clés du tableau.

Syntaxe	`int extract(array $tableau [, int $modeExtraction [, string $prefixe]])`
`$tableau`	Tableau (associatif) dont on veut se servir pour créer des variables.
`$modeExtraction`	Argument optionnel indiquant quelle stratégie adopter, principalement si la variable que l'on souhaite créer existe déjà. Cet argument peut prendre une valeur parmi : `EXTR_OVERWRITE` (valeur par défaut) remplace la valeur de la variable par la valeur trouvée dans le tableau si la clé correspond à une variable existante. `EXTR_SKIP` ignore l'élément rencontré si la clé correspond à une variable existante. `EXTR_PREFIX_SAME` fait précéder le nom de la variable par `$prefixe` si la clé correspond à une variable existante. `EXTR_PREFIX_ALL` fait systématiquement précéder le nom de la variable par `$prefixe`. `EXTR_PREFIX_INVALID` fait précéder le nom de la variable par `$prefixe` si la clé ne permet pas de créer un nom de variable valide (par exemple si la clé est en fait un index).
`$prefixe`	Argument optionnel à préciser si `$modeExtraction` prend l'une des valeurs suivantes : `EXTR_PREFIX_SAME`, `EXTR_PREFIX_ALL`, `EXTR_PREFIX_INVALID`.
`retour`	Nombre de variables générées.

compact()

Retourne un tableau associatif créé à partir des noms de variables (qui seront les clés du tableau) et de leurs valeurs.

Syntaxe	`array compact(string $variable1 [, string $variable2, ...])`
`$variable1, ...`	Noms des variables à ajouter au tableau. Ces arguments peuvent, en fait, également être des tableaux contenant des chaînes de caractères précisant des noms de variables (ou des tableaux). Chaque argument sous forme de tableau est traité de façon récursive.
`retour`	Tableau associatif contenant l'ensemble des paires (clé, valeur) correspondant aux noms de variables donnés.

Fonctions de parcours de tableau

Les tableaux étant en fait des listes de couples (clé, valeur) ou (index, valeur), ils peuvent être parcourus non pas uniquement en s'appuyant sur les clés ou les index, mais également grâce à un pointeur se déplaçant dans la liste.

current()

Retourne la valeur actuellement pointée par le pointeur interne du tableau (sans avancer le pointeur).

Syntaxe	`mixed current(array $tableau)`
`$tableau`	Tableau à parcourir.
`retour`	Valeur actuellement pointée par le pointeur interne de `$tableau`. `FALSE` si le pointeur est en dehors du tableau.

pos()

Équivalent de `current()`.

key()

Retourne la clé (ou index) de l'élément actuellement pointé par le pointeur interne du tableau (sans avancer le pointeur).

Syntaxe	`mixed key(array $tableau)`
`$tableau`	Tableau à parcourir.
`retour`	Clé (ou index) de l'élément actuellement pointé par le pointeur interne de `$tableau`. `FALSE` si le pointeur est en dehors du tableau.

reset()

Remet le pointeur interne au début du tableau et retourne le premier élément.

Syntaxe	`mixed reset(array $tableau)`
`$tableau`	Tableau à parcourir.
`retour`	Premier élément du tableau. `FALSE` si le tableau est vide.

next()

Avance le pointeur interne du tableau et retourne le nouvel élément pointé.

Syntaxe	`mixed next(array $tableau)`
`$tableau`	Tableau à parcourir.
`retour`	Élément suivant du tableau. `FALSE` si l'on a dépassé la fin du tableau.

Listing 3.13 : array_next.php

```
<?php
    // Exemple de parcours d'un tableau
    // avec le pointeur interne et la fonction
    // next();

    $tableau = array( "Début", "", "Milieu", 0, "Fin");

    // Ici, l'appel à reset() n'est pas nécessaire
    // puisque après initialisation le pointeur
    // de tableau est au debut.
    $valeur = reset($tableau);

    // Tant que valeur est différente de FALSE
    // ATTENTION:
    // 1 - Bien prendre soin de mettre les 2 signes =
    //     afin de comparer également les types
    //     car 0 == FALSE (même valeur)
    //     alors que 0 !== FALSE (même valeur mais pas même type)
    // 2 - Ce type de parcours n'est valable
    //     que si le tableau ne possède aucun
    //     élément valant FALSE
    while ($valeur !== FALSE) {
        echo "$valeur<br />";
        $valeur = next($tableau);
    }
?>
```

aura pour effet d'afficher :

Début

```
Milieu
0
Fin
```


prev()

Recule le pointeur interne du tableau et retourne le nouvel élément pointé.

Syntaxe	`mixed prev(array $tableau)`
`$tableau`	Tableau à parcourir.
`retour`	Élément précédent du tableau. `FALSE` si le début du tableau a été déplacé.

end()

Place le pointeur interne à la fin du tableau et retourne le dernier élément.

Syntaxe	`mixed end(array $tableau)`
`$tableau`	Tableau à parcourir.
`retour`	Dernier élément du tableau. `FALSE` si le tableau est vide.

each()

Retourne un tableau contenant la clé et la valeur de l'élément actuellement pointé par le pointeur interne du tableau, puis avance le pointeur.

Syntaxe	`array each(array $tableau)`
`$tableau`	Tableau à parcourir.
`retour`	Tableau composé de quatre éléments. À l'élément d'index `0` et à l'élément de clé "`key`" est associée la clé (ou l'index) de l'élément pointé. À l'élément d'index `1` et à l'élément de clé "`value`" est associée la valeur de l'élément pointé. `FALSE` si la fin du tableau est dépassée.


REMARQUE

each() et list() un duo très apprécié en concurrence avec foreach

Le tableau issu de l'appel à `each()` est souvent associé à `list()` afin de manipuler de simples chaînes plutôt que des tableaux selon le shéma `list($cle, $valeur) = each ($tableau);`. Ainsi, les tableaux sont souvent parcourus de la façon suivante :


REMARQUE

```
reset($tableau);
while (list($cle, $valeur) = each($tableau)) {
    echo "A la clé $cle est associé la valeur $valeur<br />";
}
```

Une autre solution consiste à utiliser `foreach`*. Mais il est également possible de le faire de la façon suivante :*

```
foreach($tableau as $cle => $valeur)
    echo "A la clé $cle est associée la valeur $valeur<br />";
```


Fonctions de gestion de piles

Il est possible d'utiliser les tableaux PHP comme des piles, c'est-à-dire comme des listes dans lesquelles s'empilent les éléments et auxquelles on peut accéder en prenant le premier ou le dernier élément.

array_push()

Ajoute un ou plusieurs éléments à la fin du tableau (au-dessus de la pile).

Syntaxe	`int array_push(array $tableau, mixed $valeur1, [mixed $valeur2, ...])`
`$tableau`	Tableau auquel vous souhaitez ajouter des éléments.
`$valeur1, ...`	Valeurs (ou données) que vous souhaitez ajouter au tableau.
`retour`	Retourne le nombre d'éléments du tableau après ajout.


REMARQUE

Équivalent de array_push()
À la valeur retour près, `array_push($tableau, $valeur1)` *équivaut à* `$tableau[] = $valeur1;`

array_pop()

Retourne et supprime le dernier élément du tableau (celui du dessus de la pile).

Syntaxe	`mixed array_pop(array $tableau)`
`$tableau`	Tableau que l'on veut "dépiler".
`retour`	Le dernier élément du tableau ou `NULL` si le tableau est vide (ou si l'argument n'est pas un tableau).

array_unshift()

Ajoute un ou plusieurs éléments au début du tableau (au bas de la pile).

Syntaxe	`int array_unshift(array $tableau, mixed $valeur1, [mixed $valeur2, ...])`
`$tableau`	Tableau auquel vous souhaitez ajouter des éléments.
`$valeur1, ...`	Valeurs (ou données) que vous souhaitez ajouter au tableau. La première valeur donnée correspondra à la première valeur du tableau résultat.
`retour`	Retourne le nombre d'éléments du tableau après ajout.

array_shift()

Retourne et supprime le premier élément du tableau (celui du bas de la pile).

Syntaxe	`mixed array_shift(array $tableau)`
`$tableau`	Tableau que l'on veut "dépiler" par le bas.
`retour`	Le premier élément du tableau ou `NULL` si le tableau est vide (ou si l'argument n'est pas un tableau).

Fonctions de tri

Tri inverse

array_reverse()

Retourne un tableau avec les valeurs dans l'ordre inverse de celui spécifié.

Syntaxe	`array array_reverse(array $tableau, [boolean $avecCle])`
`$tableau`	Tableau de référence.
`$avecCle`	Argument optionnel à positionner à `TRUE` si l'association clé->valeur doit être conservée (pour un tableau indexé, les valeurs conserveront également leur index). Par défaut `$avecCle = FALSE` ; le tableau résultat est alors obligatoirement un tableau indexé entre 0 et la taille du tableau -1.
`retour`	Retourne le tableau spécifié avec les valeurs classées dans l'ordre inverse.

Tri selon l'ordre croissant des valeurs

sort()

Trie les éléments d'un tableau dans l'ordre croissant des valeurs. Les clés (ou index) ne sont pas conservées ; le tableau devient obligatoirement un tableau indexé entre `0` et la taille du tableau-1.

Syntaxe	`void sort(array $tableau)`
`$tableau`	Tableau à trier.

Listing 3.14 : array_sort.php

```
<?php
    $tableauVille = array("Circuit" => "Le Mans", "Braderie" => "Lille",
                          "Horloge" => "Rouen", "Place Stanislas" => "Nancy");
    sort($tableauVille);
    for ($i=0; $i<count($tableauVille); $i++) {
        echo $i." - ".$tableauVille[$i]."<br />";
    }
?>
```

affichera

```
0 - Le Mans
1 - Lille
2 - Nancy
3 - Rouen
```

asort()

Trie les éléments d'un tableau dans l'ordre croissant des valeurs, tout en conservant l'association clé => valeur.

Syntaxe	`void asort(array $tableau)`
`$tableau`	Tableau à trier.

natsort()

Trie les éléments d'un tableau dans l'ordre croissant, dit "naturel", des valeurs. Ce tri ne se contente pas de comparer les chaînes de caractères, caractère par caractère, mais distingue les parties numériques des parties alphabétiques. Ainsi, dans l'ordre "naturel", "mot2" apparaît

avant "mot10". Les clés ou index ne sont pas conservés ; le tableau devient obligatoirement un tableau indexé entre 0 et la taille du tableau-1.

Syntaxe	`void natsort(array $tableau)`
`$tableau`	Tableau à trier.

natcasesort()

Trie les éléments d'un tableau dans l'ordre croissant, dit "naturel", des valeurs sans tenir compte de la casse. Ce tri ne se contente pas de comparer les chaînes de caractères, caractère par caractère, mais distingue les parties numériques des parties alphabétiques. Ainsi, dans l'ordre "naturel", "mot2" apparaît avant "mot10". Les clés ou index ne sont pas conservés ; le tableau devient obligatoirement un tableau indexé entre 0 et la taille du tableau -1.

Syntaxe	`void natcasesort(array $tableau)`
`$tableau`	Tableau à trier.

natsort(), natcasesort() et ordre décroissant
Les fonctions `natsort()` *et* `natcasesort()` *ne proposent pas d'équivalent pour un tri décroissant. Pour un tel tri, vous devrez faire appel à* `natsort()` *ou* `natcasesort()` *puis à* `$tableau = array_reverse($tableau);`.

Tri selon l'ordre décroissant des valeurs

rsort()

Trie les tableaux dans l'ordre décroissant des valeurs. Les clés (ou index) ne sont pas conservées ; le tableau devient obligatoirement un tableau indexé entre 0 et la taille du tableau-1.

Syntaxe	`void rsort(array $tableau)`
`$tableau`	Tableau à trier.

Voir l'exemple associé à la fonction `sort()`.

arsort()

Trie les éléments d'un tableau dans l'ordre décroissant des valeurs, tout en conservant l'association clé => valeur.

Syntaxe	`void arsort(array $tableau)`
`$tableau`	Tableau à trier.

Tri personnalisé des valeurs

usort()

Trie les éléments d'un tableau dans l'ordre croissant des valeurs, en déterminant l'ordre des valeurs à partir d'une fonction définie par l'utilisateur. Les clés (ou index) ne sont pas conservées ; le tableau devient obligatoirement un tableau indexé entre 0 et la taille du tableau-1.

Syntaxe	`boolean usort(array $tableau, fonction $fonctionComparaison)`
`$tableau`	Tableau à trier.
`$fonctionComparaison`	Nom de la fonction à utiliser pour comparer les valeurs. Cette fonction doit accepter deux arguments et retourner : `0` en cas d'égalité, `1` en cas de supériorité du premier argument sur le deuxième et `-1` sinon.
`boolean`	`TRUE`.

Listing 3.15 : array_usort.php

```
<?php
    function maFonction($valeur1, $valeur2) {
        // La fonction suivante retourne
        // un pourcentage de similarité entre
        // les 2 chaînes de caractères.
        // On pourra ainsi dire( )laquelle des 2 chaînes
        // des 2 valeurs se rapproche le plus
        // de "Etretat"

        $s1 = similar_text($valeur1, "Etretat");
        $s2 = similar_text($valeur2, "Etretat");

        if ($s1 == $s2) return 0;
        return ($s1 > $s2) ? -1 : 1;
    }

    $tableauMot = array("Etat", "Etretat", "Etretot", "Arrete");

    // Tri du plus proche du mot "Etretat"
    // au plus éloigné.
    usort($tableauMot, "maFonction");
    print_r($tableauMot);
?>
```

retourne bien les éléments du tableau du plus proche du mot "Etretat" au plus éloigné.

Array ([0] => Etretat [1] => Etretot [2] => Etat [3] => Arrete)


REMARQUE

Cas d'utilisation
Ce type de fonction est particulièrement utile dans le cas d'une méthode de comparaison un peu complexe (comme lorsque, dans l'exemple, il s'agit de calculer la similarité entre chaînes de caractères), et surtout lorsque les éléments du tableau sont des objets et que les comparaisons s'appliquent sur les attributs de l'objet.

uasort()

Trie les éléments d'un tableau dans l'ordre croissant des valeurs, en déterminant l'ordre des valeurs à partir d'une fonction définie par l'utilisateur. Les clés (ou index) ne sont pas conservées ; le tableau devient obligatoirement un tableau indexé entre 0 et la taille du tableau-1.

Syntaxe	`boolean uasort(array $tableau, fonction $fonctionComparaison)`
`$tableau`	Tableau à trier.
`$fonctionComparaison`	Nom de la fonction à utiliser pour comparer les valeurs. Cette fonction doit accepter deux arguments et retourner : `0` en cas d'égalité, `1` en cas de supériorité du premier argument sur le deuxième et `-1` sinon.
`retour`	`TRUE`.

Tri selon l'ordre croissant des clés

ksort()

Trie les éléments d'un tableau dans l'ordre croissant des clés, tout en conservant l'association clé => valeur.

Syntaxe	`boolean ksort(array $tableau)`
`$tableau`	Tableau à trier.
`retour`	`TRUE`.

Tri selon l'ordre décroissant des clés

krsort()

Trie les tableaux dans l'ordre décroissant des clés, tout en conservant l'association clé => valeur.

Syntaxe	`boolean krsort(array $tableau)`
`$tableau`	Tableau à trier.
`retour`	`TRUE`.

Tri personnalisé des clés

uksort()

Trie les éléments d'un tableau dans l'ordre croissant des clés, en déterminant l'ordre des clés à partir d'une fonction définie par l'utilisateur.

Syntaxe	`boolean uksort(array $tableau, fonction $fonctionComparaison)`
`$tableau`	Tableau à trier.
`$fonctionComparaison`	Nom de la fonction à utiliser pour comparer les clés. Cette fonction doit accepter deux arguments et retourner : `0` en cas d'égalité, `1` en cas de supériorité du premier argument sur le deuxième et `-1` sinon.
`retour`	`TRUE`.

Voir l'exemple associé à la fonction `usort()`.

Tri sur des tableaux liés

array_multisort()

Trie les éléments des tableaux selon l'ordre du premier tableau indiqué.

Syntaxe	`boolean array_multisort(array $tableau1 [,int $mode1] [, int $mode2] , array $tableau2 [, array $tableau3])`
`$tableau1`	Tableau à trier à prendre comme référence pour le tri.
`$mode1`	Sens de tri. `SORT_ASC` pour un tri croissant. `SORT_DESC` pour un tri décroissant.
`$mode2`	Option de tri: `SORT_REGULAR` pour une comparaison "traditionnelle" des valeurs. `SORT_NUMERIC` pour une comparaison numérique des valeurs. `SORT_STRING` pour une comparaison alphabétique des valeurs.
`$tableau2, ...`	Tableaux sur lesquels les permutations effectuées sur `$tableau1` doivent être répercutées.

Listing 3.16 : array_array_multisort.php

```
<?php

    // Exemple de 2 tableaux liés
    // Nom et Notes doivent toujours correspondre ...
    $eleves = array("Pierre", "Jean", "Henri", "Odille");
    $notes = array(12, 15, 13, 10);

    // ... et pourtant on voudrait trier
    // selon les prénoms des élèves.
    array_multisort($eleves, $notes);

    echo "<b>Elèves</b>:";
    print_r($eleves);

    echo "<br />";
    echo "<b>Notes</b>:";
    print_r($notes);

    echo "<br />";
    // ... ou encore selon l'ordre décroissant
    // des notes.
    array_multisort($notes, SORT_DESC, $eleves);

    echo "<br />";
    echo "<b>Elèves</b>:";
    print_r($eleves);

    echo "<br />";
    echo "<b>Notes</b>:";
    print_r($notes);

?>
```

affichera

```
Elèves:Array ( [0] => Henri [1] => Jean [2] => Odille [3] => Pierre )
Notes:Array ( [0] => 13 [1] => 15 [2] => 10 [3] => 12 )

Elèves:Array ( [0] => Jean [1] => Henri [2] => Pierre [3] => Odille )
Notes:Array ( [0] => 15 [1] => 13 [2] => 12 [3] => 10 )
```

Tri aléatoire

shuffle()

Trie les éléments d'un tableau dans l'ordre.... aléatoire. Autrement dit, mélange les éléments d'un tableau. Les clés (ou index) ne sont pas conservées ; le tableau devient obligatoirement un tableau indexé entre 0 et la taille du tableau -1.

Syntaxe	`boolean shuffle(array $tableau)`
`$tableau`	Tableau à mélanger.
`retour`	`TRUE`.

Fonctions de statistiques

Il existe quatre fonctions que l'on peut assimiler à des fonctions de statistiques.

array_sum()

Retourne la somme des valeurs des éléments d'un tableau (fort pratique pour calculer les moyennes).

Syntaxe	`mixed array_sum(array $tableau)`
`$tableau`	Tableau sur lequel doit s'effectuer l'opération.
`retour`	La somme des valeurs des éléments du tableau, de type entier ou réel, selon le contenu du tableau.

array_count_values()

Retourne un tableau précisant le nombre d'occurrences d'une valeur dans un tableau donné.

Syntaxe	`array array_count_values(array $tableau)`
`$tableau`	Tableau sur lequel doit s'effectuer l'opération.
`retour`	Tableau associatif ayant comme clés les valeurs du tableau sur lequel a été effectuée l'opération et comme valeur le nombre d'occurrences de la clé.

Listing 3.17 : array_array_count_values

```
<?php
    $tableauNote = array ("Pierre" => 10, "Marcel" => 12, "Franck" => 10,
                          "Jean" => 10, "Marie" => 15, "Cecile" => 12);
```

```
    $stats = array_count_values($tableauNote);

    echo "A ce devoir ...<br />\n";
    while (list($note, $nb) = each($stats)) {
        echo "$nb élèves ont eu $note<br />";
   }
?>
```

donnera le résultat

```
A ce devoir ...
3 élèves ont eu 10
2 élèves ont eu 12
1 élèves ont eu 15
```

(nous vous laissons le soin d'améliorer ce script pour traiter le cas où `$nb = 1`).

Voir aussi les fonctions `min()` et `max()` développées dans le chapitre "Les fonctions mathématiques".

Fonctions mettant en œuvre plusieurs tableaux

Comparaison de tableaux

array_diff()

Retourne un tableau présentant les valeurs contenues dans un tableau, mais absentes d'autres tableaux.

Syntaxe	`array array_diff(array $tableau1, array $tableau2, [array $tableau3, ...])`
`$tableau1`	Tableau dont on veut comparer le contenu avec ceux des autres tableaux.
`$tableau2, ...`	Tableaux dans lesquels on recherche l'éventuelle présence des valeurs du tableau 1.
`retour`	Tableau présentant toutes les valeurs contenues dans `$tableau1` et qui n'ont été trouvées dans aucun des autres tableaux précisés. Les clés sont conservées (identiques à celle de `$tableau1`).

array_intersect()

Retourne un tableau présentant les valeurs contenues dans un tableau et dans un ensemble d'autres tableaux.

Syntaxe	`array array_intersect(array $tableau1, array $tableau2, [array $tableau3, ...])`
`$tableau1`	Tableau dont on veut comparer le contenu avec ceux des autres tableaux.
`$tableau2, ...`	Tableaux dans lesquels on recherche l'éventuelle présence des valeurs du tableau 1.
`retour`	Tableau présentant toutes les valeurs contenues dans `$tableau1` et qui ont été trouvées dans tous les autres tableaux précisés. Les clés sont conservées (identiques à celle de `$tableau1`). S'il n'y a pas de valeurs communes alors c'est un tableau vide qui est retourné.

array_intersect_assoc()

Retourne un tableau présentant les valeurs et les clés associées à celle-ci, contenues dans un tableau et dans un ensemble d'autres tableaux.

Syntaxe :	`array array_intersect_assoc(array $tableau1, array $tableau2, [array $tableau3, ...])`
`$tableau1`	Tableau dont on veut comparer le contenu avec ceux des autres tableaux.
`$tableau2, ...`	Tableaux dans lesquels on recherche l'éventuelle présence des valeurs du tableau 1.
`retour`	Tableau présentant toutes les valeurs et les clefs contenues dans `$tableau1` et qui ont été trouvées dans tous les autres tableaux précisés. Les clés sont conservées (identiques à celle de `$tableau1`).

```
<?php
$tableau1 = array(0=>"Belgique", 1=>"France", 2=>"Italie", 3=>"Suisse");
$tableau2 = array(1=>"France", 3=>"Italie", 2=>"Irlande");
$tableauFinal = array_intersect_assoc($tableau1, $tableau2);
print_r($tableauFinal);
?>
Array
(
    1 => France
)
```

Fusion de tableaux

array_merge()

Retourne un tableau issu de la fusion d'un ensemble de tableaux.

Syntaxe	`array array_merge(array $tableau1, array $tableau2, [array $tableau3, ...])`
`$tableau1, ...`	Les tableaux à fusionner.
`retour`	Tableau présentant toutes les valeurs contenues dans les différents tableaux. Au fil du parcours des tableaux à fusionner, les valeurs ont été ajoutées à la fin du tableau résultat. Les valeurs liées à des clés peuvent être écrasées si la clé est rencontrée plusieurs fois.

array_merge_recursive()

Retourne un tableau issu de la fusion d'un ensemble de tableaux. Dans ce cas, si une clé donnée est rencontrée plusieurs fois, la valeur dans le tableau résultat ne sera pas la dernière valeur rencontrée, mais la fusion (dans un tableau) des valeurs rencontrées.

Syntaxe	`array array_merge_recursive(array $tableau1, array $tableau2, [array $tableau3, ...])`
`$tableau1, ...`	Les tableaux à fusionner.
`retour`	Tableau présentant toutes les valeurs contenues dans les différents tableaux. Au fil du parcours des tableaux à fusionner, les valeurs ont été ajoutées à la fin du tableau résultat. Les valeurs liées à des clés rencontrées plusieurs fois sont regroupées dans un tableau.

Listing 3.18 : array_array_merge_recursive.php

```
<?php
    $agenda1 = array("Lundi" => "Médecin", "Mardi" => "Coiffeur");
    $agenda2 = array("Mardi" => "Réunion", "Vendredi" => "VTT");

    $agenda = array_merge_recursive($agenda1, $agenda2);
    print_r($agenda);
?>
```

Voilà comment fusionner intelligemment deux agendas...

```
Array ( [Lundi] => Médecin [Mardi] => Array ( [0] => Coiffeur [1] => Réunion )
✂ [Vendredi] => VTT )
```

Fonctions de traitement personnalisé des tableaux

array_filter()

Retourne un tableau filtré à partir d'une fonction écrite par l'utilisateur.

Syntaxe	`array array_filter(array $tableau [, function $fonctionFiltre])`
`$tableau`	Tableau à filtrer.
`$fonctionFiltre`	Fonction à appeler pour déterminer si l'élément doit être conservé ou non dans le tableau résultat. Cette fonction doit prendre en compte un argument, et retourner `TRUE` si l'élément doit être conservé, `FALSE` sinon. Ce paramètre est (curieusement) optionnel. Si aucune fonction de filtrage n'est fournie, alors le tableau est restitué tel quel.
`retour`	Retourne le tableau d'entrée sans les éléments qui ont été indiqués, par la fonction `$fonctionFiltre`, comme étant à filtrer. Les clés et index du tableau sont conservés.

Listing 3.19 : array_array_filter.php

```
<?php

    // Exemple de filtre qui ne conserve que les
    // prénoms commençant par M
    function monFiltre($prenom) {
        return ($prenom[0] == "M");
    }

    $prenoms = array("Stéphane", "Marie", "Thierry", "Abi",
                     "Christophe", "Manu", "Anne");

    print_r(array_filter($prenoms , "monFiltre"));
?>
```

ne conserve bien que les prénoms commençant par M.

```
Array ( [1] => Marie [5] => Manu )
```

array_map()

Retourne un tableau issu du résultat de l'application d'une fonction écrite par l'utilisateur, et prenant en compte les données d'un ou de plusieurs tableaux.

Syntaxe	`array array_map(function $fonction, array $tableau1 [, array $tableau2, ...])`
`$fonction`	Fonction à appeler pour chaque élément des tableaux `$tableau1`, `$tableau2`, etc. Cette fonction doit prendre en compte autant d'arguments que de tableaux fournis.
`$tableau1, ...`	Tableaux contenant les valeurs sur lesquelles doit s'appliquer la fonction. Si certains tableaux sont moins longs que le plus grand des tableaux, des valeurs nulles complètent ces tableaux. Ces tableaux sont traités dans l'ordre de leurs éléments (et non de leur index).

`retour`	Tableau indexé de la taille du plus grand des tableaux passés en paramètre et ayant pour valeurs le résultat de la fonction appliquée aux éléments des tableaux passés en paramètre.

Listing 3.20 : array_array_map.php

```
<?php
    // Exemple de 2 tableaux servant
    // à stocker des coordonnées
    // C'est pas aussi beau que la programmation
    // objet mais des fois ça dépanne

    $tabX = array (1, 5, 2, 4);
    $tabY = array (10, 2, 3, 2);


    // Exemple d'une fonction permettant
    // le calcul de la distance entre un point
    // et le centre (0, 0)
    function distance($x, $y) {
        return sqrt($x*$x + $y*$y);
    }

    // Et, hop, d'un simple appel à
    // array_map j'ai mes distances
    // pour tous les points.
    print_r(array_map("distance", $tabX, $tabY));
?>
```

retournera le tableau des distances suivant :

```
Array ( [0] => 10.049875621121 [1] => 5.3851648071345 [2] => 3.605551275464 [3]
✂ => 4.4721359549996 )
```

array_reduce()

Retourne le résultat d'une opération définie par l'utilisateur, appliquée itérativement sur l'ensemble des valeurs d'un tableau.

Syntaxe	`mixed array_reduce(array $tableau, function $fonction [,int $valeurInitiale])`
`$tableau`	Tableau contenant les valeurs sur lesquelles doivent s'effectuer l'opération.
`$fonction`	Fonction à appeler itérativement sur chaque élément du tableau. Cette fonction doit prendre en compte deux arguments, le premier étant le résultat du calcul trouvé jusque-là, le second étant la valeur à prendre en compte pour la poursuite du calcul. La fonction doit retourner le nouveau résultat obtenu.

`$valeurInitiale`	Argument optionnel, précisant la valeur initiale utilisée pour le calcul. Par défaut `$valeurInitiale = 0`.
`retour`	Le résultat du calcul voire la valeur initiale si le tableau est vide.

Listing 3.21 : array_array_reduce.php

```
<?php

    // Calcul de P(x0)
    // où P polynôme
    // P(x) = x^4 + 2x^3 + x^2 + 4x + 6
    //      = (((x + 2)*x + 1)*x +4)*x + 6
    // Soit le calcul itératif
    // Resultat = 1
    // Resultat = Resultat*x0 + 2
    // Resultat = Resultat*x0 + 1
    // Resultat = Resultat*x0 + 4
    // Resultat = Resultat*x0 + 6

    // Ainsi avec le tableau des coefficients
    $tabCoef = array(2, 1, 4, 6);

    // et la ch'tite fonction
    function polynome($resultat, $coef) {
        $x0=5; // Calcul pour x0=5;
        return $resultat*$x0 + $coef;
    }

    // J'ai mon résultat
    echo "P(5)=".array_reduce($tabCoef, "polynome", 1);
?>
```

Si vous souhaitez vraiment le savoir, cela retourne :

```
P(5)=926
```

array_walk()

Appelle une fonction utilisateur sur chaque élément d'un tableau.

Syntaxe	`boolean array_walk(array $tableau, function $fonction [, mixed $argument])`
`$tableau`	Tableau contenant les valeurs sur lesquelles doivent s'effectuer les opérations.
`$fonction`	Fonction à appeler pour chaque élément du tableau. Cette fonction doit prendre en compte deux arguments (ou trois si `$argument` est spécifié), le premier argument étant la valeur de l'élément actuellement considéré,

	le second étant la clé (ou index) de l'élément actuellement considéré. Le troisième argument prendra la valeur de `$argument`). Attention ! il n'est pas possible dans ce cas d'appeler une fonction PHP, vous devez nécessairement créer votre propre fonction.
`$argument`	Argument optionnel, passé en troisième paramètre de la fonction utilisateur.
`retour`	`TRUE` si l'opération s'est bien passée, `FALSE` sinon.


REMARQUE

Passage par valeur

N'oubliez pas que, par défaut, les paramètres passés aux fonctions le sont par valeur. Si vous souhaitez utiliser `array_walk()` pour modifier les valeurs du tableau, n'oubliez pas d'utiliser une interface similaire à `mafonction(&$valeur, $clé)`.

Listing 3.22 : array_array_walk.php

```
<?php

    $menu = array ("Accueil", "Forum", "A propos");

    // Fonction pour afficher le contenu
    // du menu avec l'index et un séparateur
    function affiche($valeur, $key, $separateur) {
        echo ($key+1)." $separateur ".$valeur."<br />";
    }

    // Fonction pour modifier les étiquettes
    // du menu
    function patch(&$valeur, $key) {
        $valeur="[".$valeur."]";
    }

    // Affiche le menu
    array_walk($menu, "affiche", "-");

    // Patche le menu
    array_walk($menu, "patch");

    // ce qui donne le résultat
    print_r($menu);
?>
```

retournera

```
1 - Accueil
2 - Forum
3 - A propos
Array ( [0] => [Accueil] [1] => [Forum] [2] => [A propos] )
```

3.9. Les inclusions de fichiers

Le langage PHP permet d'insérer, dans un script, du code qui provient d'un (ou de plusieurs) autres fichiers.

Pour cela, PHP propose quatre fonctions (ou plutôt, instructions) :

- `require();`
- `include();`

et leurs variantes, depuis PHP 4.0.1,

- `require_once();`
- `include_once().`

Ces instructions sont généralement utilisées pour intégrer des scripts dans lesquels des objets ou des fonctions réutilisables ont été définis (dans ce cas, les fichiers inclus font office de bibliothèques de fonctions), ou bien pour intégrer des portions de code contenant les paramètres du script (dans ce cas, les fichiers inclus font office de fichiers de configuration).

Listing 3.23 : include_01.php

```
<?php

    // Inclut les paramètres
    include("include_01a.php");

    // Inclut les fonctions
    include("include_01b.php");

    monEcho($maChaine);
?>
```

Listing 3.24 : include_01a.php

```
<?php
    // Exemple de fichier de configuration
    // vraiment minimaliste

    $maChaine = "Bonjour tout le monde";
?>
```

Listing 3.25 : include_01b.php

```
<?php
    // Exemple de bibliothèque de fonctions
    // encore une fois bien minimaliste

    function monEcho($chaine)
```

```
    {
        echo $chaine;
    }
?>
```

sera alors équivalent (commentaires exclus) à :

Listing 3.26 : include_01equiv.php

```
<?php

    $maChaine = "Bonjour tout le monde";

    function monEcho($chaine)
    {
        echo $chaine;
    }

    monEcho($maChaine);
?>
```


REMARQUE

Fonction ou instruction ?

`require()` et `include()` ne sont pas de véritables fonctions, et l'on peut aussi bien écrire :

- *`include("nom de fichier");`*
- *`include "nom de fichier";`*

Ces instructions peuvent être utilisées de façon conditionnelle ou bien encore dans des boucles, comme le montrent les exemples suivants :

Listing 3.27 : include_02.php

```
Je suis le script principal<br />
je vais tenter d'insérer les scripts<br />
après avoir testé leur existence<br />
<?php
    if (file_exists("include_02a.php")) {
        include("include_02a.php");
    }

    if (file_exists("fichierinconnu.php")) {
        include("fichierinconnu.php");
    }
?>
```

Listing 3.28 : include_02a.php

```
* Je suis un simple script à inclure nommé include_02a.php
```

affichera :

```
Je suis le script principal
je vais tenter d'insérer les scripts
après avoir testé leur existence
* Je suis un simple script à inclure nommé include_02a.php
```

Le tout sans générer d'erreur, puisque le script principal ne tentera jamais d'inclure le fichier inexistant.


REMARQUE

Accolades nécessaires

Notez la présence des accolades qui définissent le bloc contenant uniquement l'instruction `include`*. Ces accolades sont nécessaires, il ne faut donc pas les omettre.*

Listing 3.29 : include_03.php

```
Je suis le script principal<br />
je vais tenter d'insérer un script en boucle<br />
<?php
    for ($i=0; $i<3; $i++) {
        include("include_03a.php");
    }
?>
```

Listing 3.30 : include_03a.php

```
* Je suis un simple script à inclure nommé include_03a.php<br />
```

affichera, quant à lui :

```
Je suis le script principal
je vais tenter d'insérer un script en boucle
après avoir testé leur existence
* Je suis un simple script à inclure nommé include_03a.php
* Je suis un simple script à inclure nommé include_03a.php
* Je suis un simple script à inclure nommé include_03a.php
```


REMARQUE

Évolution de l'instruction require()

Même si cela n'était pas vrai auparavant pour l'instruction `require()`*, depuis la version 4.0.2, les fonctions* `require()` *et* `include()` *se comportent sensiblement de la même façon (la commande* `require()` *s'étant ralliée au comportement de la commande* `include()`*).*

Comme cela se devine dans les exemples précédents, les scripts insérés commencent sur la base d'un texte brut (ou code HTML) ; il faudra donc rouvrir les balises PHP pour les portions de code PHP. De ce fait, les scripts insérés sont similaires aux scripts principaux.

Inclusions multiples

Il peut arriver que, sans véritablement le vouloir, un script soit inclus plusieurs fois. Cela arrive notamment lorsqu'un script A inclut un script B et un script C, alors que le script B lui-même inclut le script C (c'est généralement ce qui arrive si les règles de codage ne sont pas suffisamment claires sur ce point).

Si le script inclus de multiples fois contient des déclarations de fonctions (ou d'objets) alors, inévitablement, vous vous retrouverez avec un message d'erreur indiquant que la fonction (ou l'objet) a déjà été définie. Afin de vous affranchir de ce problème, vous pouvez faire appel à l'instruction `include_once()` ou `require_once()` afin de n'inclure le script qu'au premier appel.

Ainsi, si l'on reprend l'exemple précédent avec `include_once()`, cela donnera :

Listing 3.31 : include_04.php

```
Je suis le script principal<br />
je vais tenter d'insérer un script en boucle<br />
<?php
    for ($i=0; $i<3; $i++) {
        include_once("include_03a.php");
    }
?>
```

et affichera seulement :

```
Je suis le script principal
je vais tenter d'insérer un script en boucle
après avoir testé leur existence
* Je suis un simple script à inclure nommé include_03a.php
```

Notez, à cette occasion, qu'il est à tout moment possible de connaître la liste des fichiers inclus grâce à la fonction `get_included_files()` (ou son alias `get_required_files()`).

get_included_files()

Retourne la liste des fichiers inclus.

Syntaxe	`array get_included_files(void)`
`retour`	Tableau indexé contenant la liste des noms complets (chemins absolus) des fichiers inclus par `include()`, `require()`, `include_once()` ou `require_once()`. Les noms des fichiers inclus plusieurs fois n'apparaissent qu'une fois.

Les noms des fichiers inclus

Les noms des fichiers inclus peuvent porter n'importe quelle extension. Mais, dans la pratique, il faut être prudent et bien choisir son extension. Nous vous conseillons une extension "*_inc.php*".


ATTENTION

Extension des fichiers inclus et risques de piratage

Si vous ne donnez pas l'extension .php, les fichiers inclus, s'ils sont accédés directement, ne seront pas interprétés et livreront tous leurs secrets.

En effet, si, par exemple, votre script principal inclut un fichier nommé config.in" et qu'un pirate connaît l'existence de ce fichier, il pourra l'appeler directement depuis son navigateur et lire simplement son contenu, comme pour tout bon fichier texte (et il y trouvera certainement des choses intéressantes du genre `$motdepasse = 'secret';`*). S'il s'agit d'un fichier contenant des fonctions ou des classes, le pirate pourra récupérer votre code source (et votre savoir-faire). Et pourtant, le moyen de se protéger est très simple : si vous ajoutez simplement l'extension .php à vos scripts inclus, le code PHP, avant d'être envoyé au navigateur du pirate, sera interprété, et le pirate ne pourra lire que le résultat de l'interprétation (généralement un simple page blanche, puisque par exemple* `$motdepasse = 'secret';` *n'affichera rien).*

Les fichiers insérés distants

Les fichiers à insérer sont généralement sur le même serveur que le script principal, mais il est également possible d'insérer un fichier situé sur un autre serveur.

Pour cela, il suffit de préciser l'URL complète du fichier, par exemple **http://www.php.net/chemin/fichier.html**, à condition toutefois que la directive de compilation `--disable-url-fopen-wrapper` n'ait pas été utilisée. Il est à noter que, dans ce cas, si vous appelez un script PHP, seul le résultat de l'interprétation du script par le serveur distant sera inclus dans le script principal (et non le code source du script inclus).


RENVOI

Vous pouvez vous reporter au chapitre "Prise en main" pour plus de détails sur la compilation de PHP.


REMARQUE

Ouf ! Nous sommes protégés

La remarque précédente est importante. Elle implique que, tant que vous utilisez l'extension .php et que votre serveur interprète les fichiers .php, un pirate ne peut pas espérer lire vos fichiers de configuration par un simple `include("http://www.serveurquejaimeraispirater.com/fichierconfig.php");`.

Le passage de paramètres

Les fichiers étant tout bêtement intégrés dans le script principal, les variables globales qui auront été définies avant les commandes `include()` ou `require()` seront accessibles par les scripts intégrés. Et, réciproquement, les variables globales définies (ou modifiées) dans les fichiers inclus (interprétés localement) seront accessibles par les fichiers inclus suivants ainsi que par le script principal.

Ainsi, si le script inclus nécessite un certain nombre de paramètres, il n'est pas nécessaire (et cela ne fonctionnera pas) de faire :

```
<?php
    include("fichier_inc.php?parametre1=valeur1&parametre2=valeur2")
?>
```

mais plutôt :

```
<?php
    $parametre1 = valeur1;
    $parametre2 = valeur2;
    include("fichier_inc.php");
?>
```

Le seul cas où, dans un `include()`, on peut être amené à passer des paramètres, c'est celui où le fichier inclus est un script PHP distant qui nécessite des paramètres. C'est un cas bien particulier, puisqu'alors le script inclus sera interprété par le serveur distant avant d'être inclus (et non inclus tel quel dans le script principal).

```
<?php
    include("http://www.php.net/script.php?param1=valeur1");
?>
```

Les chemins relatifs

Les chemins précisés dans ces commandes, s'ils ne sont pas absolus, sont toujours relatifs au script en cours d'exécution. Cela peut, à première vue, paraître banal, mais, c'est en fait lourd de conséquences.

Si votre script *principal.php* du projet `projet` est stocké dans un répertoire *projet* et qu'il inclut un script *inclus1_inc.php* de la bibliothèque `biblio` stocké dans un répertoire *biblio*, vous serez tenté de mettre la commande `include("../biblio/inclus1_inc.php")`. À vrai dire, jusque-là, tout va bien.

Mais si votre script *inclus1_inc.php* a besoin d'un autre fichier *inclus2_inc.php* (disons) de la même bibliothèque, ce dernier aura probablement utilisé la commande `include("inclus2_inc.php")`. Et c'est là le problème.

Si vous décidez d'exécuter le script *inclus1_inc.php*, cela fonctionnera parfaitement. Mais, pour le script *principal.php* le chemin précisé dans la commande `include` de *inclus1_inc.php* sera relatif au script en cours (relatif au répertoire *projet*). Autrement dit, on va chercher *inclus2_inc .php* sous *projet* et non sous *../biblio*, ce qui, de toute évidence, ne fonctionnera pas.

Cela voudrait dire que, pour corriger le problème, nous devrions remplacer le chemin précisé dans *inclus1_inc.php* par un chemin relatif à *projet*. En d'autre termes, remplacer `include("inclus2_inc.php")` par `include("../biblio/inclus2_inc.php")`. Toutefois, cela est insatisfaisant à plus d'un titre.

1. À la lecture de la bibliothèque `biblio` cette référence *../biblio* est plutôt incongrue.
2. Cette solution ne serait pas satisfaisante pour un projet situé dans un autre niveau d'arborescence.
3. L'appel direct *inclus1_inc.php* fonctionne toujours, mais uniquement parce que nous avons pris un cas de figure plutôt simple (*projet* et *biblio* sont au même niveau d'arborescence).

Pour pallier ce problème, nous avons deux solutions possibles :

- Modifier le paramètre `include_path`.
- Toujours utiliser des chemins absolus.

Le paramètre `include_path` est un paramètre du fichier *php.ini* qui indique où aller chercher le fichier à inclure. Ce paramètre définit une liste de chemins séparés par des `:` sous Linux/UNIX, mais par des `;` sous Windows. Dans ce cas, il vous faudra modifier le fichier *php.ini* pour chaque bibliothèque ajoutée (et pour chacun des serveurs).


RENVOI

Vous pouvez vous reporter au chapitre "Prise en main" pour plus de détails sur le fichier php.ini.

Ce paramètre `include_path` peut toutefois être modifié au niveau du script (avant de faire l'appel à `include()` ou `require()`) grâce à la fonction `ini_set()`, selon le modèle `ini_set("include_path","<nouvelle liste de répertoires>");`.


ASTUCE

Construction automatique de chemins absolus

À première vue, l'idée de toujours préciser des chemins absolus lorsque l'on désire inclure des fichiers peut sembler extrêmement contraignante. En effet, cela signifie, en théorie, que si l'on souhaite déplacer l'espace hébergeant l'ensemble des scripts (ou installer les scripts sur une autre machine proposant une autre arborescence), tous les chemins seront à modifier. Mais heureusement, en pratique, il est possible de construire dynamiquement ces chemins absolus.

En effet, la constante `__FILE__` retourne le chemin absolu et le nom du fichier inclus. Le chemin absolu du fichier ayant un chemin relatif (au fichier inclus) cheminRelatif/fichier.php sera alors :

```
dirname(__FILE__)."/cheminRelatif/fichier.php"
```

Il suffit donc de remplacer l'intuitif

```
include("cheminRelatif/fichier.php");
```

par

```
include(dirname(__FILE__)."/cheminRelatif/fichier.php").
```

Cas d'erreur et code retour

Si le script à inclure n'est pas trouvé alors :

- L'instruction `include()` générera un message d'erreur de type "alerte" (`WARNING`) et le déroulement du script principal se poursuivra.
- L'instruction `require()` générera un message d'erreur de type "fatale" (`FATAL`) et le déroulement du script principal s'interrompra.

Si une erreur est levée (même une erreur de type "fatale") dans un script inclus, cela n'interrompt pas pour autant l'exécution du script principal (que ce soit avec `include()` ou `require()`).

L'instruction `include()` retourne un code d'erreur, ce qui n'est pas le cas de l'instruction `require()`.

Par défaut, le code retour de l'instruction `include()` est : `TRUE` en cas de succès et `FALSE` en cas d'échec. Mais il est également possible de générer son propre code retour (en cas de succès de l'inclusion) en sortant du script inclus par l'instruction `return()` suivie de la valeur à retourner (comme nous le ferions pour une simple fonction).

Listing 3.32 : include_05.php

```
<?php
    $retour = include("include_05_inc.php");
    echo "Code retourné par le fichier inclus = $retour<br />";
?>
```

Listing 3.33 : include_05_inc.php

```
<?php
    echo "* Je suis le fichier inclus,<br />";
    echo "je me contente d'afficher ce message<br />";
    echo "et de retourner 5<br />";

    return 5;
?>
```

affichera donc :

```
* Je suis le fichier inclus,
je me contente d'afficher ce message
et de retourner 5
Code retourné par le fichier inclus = 5
```

Différences entre include() et require()
Depuis la version 4.0.2, les instructions `include()` et `require()` ne diffèrent que sur deux des points évoqués dans ce chapitre :

- *Comportement en cas d'inexistence du fichier à inclure ;*
- *Présence ou non d'un code retour.*

Un cas d'utilisation pratique mais potentiellement dangereux

La fonction `include()` est souvent utilisée pour émuler l'utilisation de frames sans faire appel à la balise HTML `<frame>`. Dans ce cas, la page principale du site contient, généralement, un en-tête, un ou des menus (dans un bandeau supérieur, gauche ou droit), le contenu et, éventuellement, une ligne de copyright. Le tout pouvant prendre la forme d'un tableau, où , quasiment, seule la partie "contenu" change (d'une page à l'autre). L'idée consiste alors à fournir au script principal un paramètre indiquant quelle page de contenu afficher (techniquement, cela consiste en un simple `include`). Ce qui donne à peu de choses près (je vous laisse fignoler) le code suivant :

Listing 3.34 : include_frame01.php

```
<html>
<body>
<table width="100%">
<tr><td colspan="3" align="center">Entête</td></tr>
<tr><td>Menu Gauche</td>
    <td>
        <?php
            include($_GET["url"]);
 ?>
    </td>
    <td>Menu Droit</td>
</tr>
<tr><td colspan="3" align="center">Copyright</td></tr>
</table>
</body>
</html>
```

qui pourra être appelé comme suit :

```
include_frame01.php?url="accueil.html"
```

```
include_frame01.php?url="mavie.html'
```

Pourquoi éviter la balise <frame> ?

Le problème que l'on peut rencontrer en utilisant des frames HTML est lié au fait que ce qui apparaît dans le navigateur n'est pas le résultat de l'interprétation d'une seule URL mais de plusieurs (celle qui définit les frames et celles qui définissent leur contenu). Ainsi, les moteurs de recherche, lorsqu'ils indexent des sites contenant des frames, ne font pas de distinction entre ces différentes URL – ce qui bien souvent les amène à proposer une réorientation vers une des frames hors de son contexte, masquant généralement la frame qui contient le menu principal, et empêchant ainsi la navigation sur le site. Il existe de nombreux moyens de contourner ce problème : l'utilisation du Javascript en est un, mais l'idéal est certainement de n'avoir qu'une page unique.

Cependant, ce type d'implémentation pose de sérieux problèmes de sécurité. En effet, rien n'empêche un pirate d'appeler lui-même

include_frame01.php?url=http://www.sitedupirate.com/scriptmechant.txt.

Et si *scriptmechant.txt* affiche (mais n'exécute pas) un texte contenant un script PHP, ce script sera exécuté sur VOTRE serveur au moment de l'`include`. Le pirate pourra alors faire ce que bon lui semble sur VOTRE serveur, à commencer par exécuter une commande afin d'archiver du code source de votre site (et récupérer ainsi vos mots de passe d'accès à la base de données).

Dans ce cas de figure, la première précaution à prendre consiste donc à empêcher l'inclusion de fichiers distants. Une solution simple et doublement utile consiste à faire précéder la commande `include()` d'un test d'existence du fichier par `file_exists()`. Cela vous évitera, d'une part, de tenter d'inclure des fichiers qui n'existent pas et, d'autre part, comme `file_exists()` retourne `FALSE` dans le cas de fichiers distants, d'inclure de tels fichiers.

Listing 3.35 : include_frame02.php

```
<html>
<body>
<table width="100%">
<tr><td colspan="3" align="center">Entête</td></tr>
<tr><td>Menu Gauche</td>
    <td>
        <?php
            if (file_exists($_GET["url"])) include($_GET["url"]);
 ?>
    </td>
    <td>Menu Droit</td>
</tr>
<tr><td colspan="3" align="center">Copyright</td></tr>
</table>
</body>
</html>
```

Vous voilà rassuré ?

Eh bien, vous avez tort !

Ce n'est pas parce que le fichier est sur votre serveur et qu'il existe (et qu'il est accessible depuis votre compte) que vous êtes à l'abri. Eh oui, les pirates sont de petits malins (c'est d'ailleurs bien souvent par défi que les pirates sévissent). La plupart des serveurs web sont configurés pour tracer les événements (à titre statistique ou de surveillance). Avec Apache, les fichiers de traces sont, par défaut, *access_log* et *error_log* : ils contiennent généralement (selon la configuration) l'URL de la page demandée, l'adresse IP du client, etc. L'un est destiné aux accès réussis, l'autre aux accès ayant échoués. Évidemment, ces fichiers sont lisibles par le serveur et un appel à `file_exists()` sur ces fichiers retournera `TRUE`. Si le pirate fait pointer `$url` sur ces fichiers, ils seront donc inclus. Vous me direz "oui, mais qu'est-ce que ça va lui apporter de lire le contenu de ce fichier ?". À première vue rien, sinon que... je vous l'ai dit, ils sont malins !. Il lui suffit de trouver le moyen d'inclure dans ces fichiers (et plutôt dans le fichier *error_log*) le code PHP qu'il souhaite exécuter. Pour cela, il suffit qu'il appelle une URL bidon dans laquelle il intègre le code (ex. : **http://www.siteattaque.com/<moncode>**). Et voilà, le tour est joué !

Conclusion, il faut également s'assurer que le fichier inclus ne pourra être, d'une manière ou d'une autre, un fichier altéré par un pirate (ex. : contenu d'un livre d'or s'il est stocké dans un fichier et non dans une base de données).

Si vous ne créez aucun fichier (style livre d'or) dans l'espace de votre serveur, vous pouvez alors certainement vous protéger de ce type d'attaque en remplaçant le test précédant par

```
if (eregi("^".$_SERVER["DOCUMENT_ROOT"], realPath($url))) ...
```

qui vérifiera que le chemin réel de l'URL spécifiée commence bien par `$_SERVER["DOCUMENT_ROOT"]` (la racine du serveur web).

Les pirates
N'oubliez pas : il y a probablement plus de mauvais administrateurs réseaux et de mauvais programmeurs que de pirates mal intentionnés. De plus, les attaques que l'on peut subir apprennent généralement plus qu'elles ne causent de désagréments (hormis une grosse perte de temps).

3.10. Les classes, les objets

Avec la version 5, PHP se tourne un peu plus vers le monde des langages orientés objets (notez l'importance, ici, du terme "orienté"). Ce n'est toutefois pas un langage à objets comme peut l'"être Eiffel et contrairement aux autres langages orientés objets, tels que C++ ou Java, PHP reste un langage de script.

Le langage Eiffel
Créé par Bertrand Meyer, le langage Eiffel a trouvé sa place dans le milieu universitaire en tant qu'outil d'enseignement de la "philosophie" objet des langages de programmation. Son typage très fort le rend particulièrement intransigeant.

Avant d'expliquer comment cela se gère au niveau de PHP, il peut être intéressant de rappeler ce qu'est la programmation objet.

L'idée est d'avoir un langage de programmation où les données d'une même nature et les fonctions qui permettent de les manipuler sont intimement liées au sien d'entités appelées objets. Les classes associées à ces objets auront une fonction d'initialisation (appelée "constructeur"), des attributs et des fonctions (propriétés) qui leur sont propres (appelés méthodes). Toutes ces entités répondront aux exigences de l'objet.

Prenons un exemple simple : une boutique d'informatique. On pourra définir un ordinateur à partir de plusieurs classes :

- Une classe `Ordinateur` (on appellera ordinateur l'association d'un boîtier, d'un écran, d'un clavier et d'une souris) ;
- Une classe `Boitier` (on estime qu'un boîtier est déjà équipé) ;
- Une classe `Ecran` ;

- Une classe `Clavier` ;
- Une classe `Souris`.

Chacune de ces classes comportera un certain nombre d'attributs et de méthodes :

- `Souris` : une souris sera définie par son `prix` et son `modele`. On ajoutera deux fonctions permettant de modifier le prix et le modèle : `modifierPrix()`, `modifierModele()`.
- `Clavier` : dans un premier temps, nous nous contenterons des mêmes informations que celles définies pour la classe `Souris`.
- `Ecran` : idem.
- `Boitier` : idem mais nous ajouterons un attribut `details` et la fonction `modifierDetails()`.
- `Ordinateur` : on peut associer un identifiant unique à un ordinateur `identifiant`, et une méthode pour calculer son prix `calculerPrix()` en fonction des éléments qui le constitue. Quatre attributs pourront définir chacune des pièces d'un ordinateur : `boitier`, `ecran`, `clavier`, `souris`.

Alors que la plupart des attributs seront de type simple: réel (pour le prix), chaîne de caractères (pour le modèle); La classe `Ordinateur`, elle, utilise des attributs `boitier`, `ecran`, `clavier`, `souris` qui sont des types complexes: à savoir les classes précédemment définies.

La création d'un objet (instanciation d'une classe) passe par l'appel d'une méthode particulière appelée constructeur. Le constructeur des classes `Souris`, `Clavier` et `Ecran` aura comme paramètres le prix et le modèle. Le constructeur de la classe `Boitier` aura comme paramètres le prix, le modèle et une description. La classe `Ordinateur` aura un constructeur ayant comme attributs un boîtier, un écran, un clavier et une souris.

Ce qui donne en notation "abstraite" :

Classe Ordinateur

Constructeur :

- ordinateur (boîtier : BOITIER, écran : ECRAN, clavier : CLAVIER, souris : SOURIS).

Attributs :

- `boitier` : BOITIER ;
- `ecran` : ECRAN ;
- `clavier` : CLAVIER ;
- `souris` : SOURIS.

Fonctions :

- `calculerPrix()` : REAL

Classe Boitier

Constructeur :

- boîtier (modele : STRING, details : STRING, prix : REAL).

Attributs :

- `modele` : STRING ;
- `details` : STRING ;
- `prix` : REAL.

Fonctions :

- `modifierModele` (modele:STRING) : VOID ;
- `modifierDetails` (details : STRING) : VOID ;
- `modifierPrix` (details:STRING) : VOID.

Classe Ecran

Constructeur :

- écran (modele : STRING, prix : REAL).

Attributs :

- `modele` : STRING ;
- `prix` : REAL.

Fonctions:

- `modifierModele` (modele : STRING) : VOID ;
- `modifierPrix` (details : STRING) : VOID.

Classe Clavier

Constructeur :

- clavier (modele : STRING, prix : REAL).

Attributs :

- `modele` : STRING ;
- `prix` : REAL.

Fonctions :

- `modifierModele` (modele : STRING) : VOID ;

- `modifierPrix` (details : STRING) : VOID.

Classe Souris

Constructeur :

- souris (modele : STRING, prix : REAL).

Attributs :

- `modele` : STRING ;
- `prix` : REAL.

Fonctions :

- `modifierModele` (modele : STRING) : VOID ;
- `modifierPrix` (details : STRING) : VOID.

Ce modèle définit simplement un ordinateur tel qu'il a été décrit.

L'avantage de ce type de programmation, c'est qu'il est très facilement modifiable et corrigible. S'il est décidé que le calcul du coût d'un ordinateur n'est plus la simple addition des pièces le composant, mais le coût des pièces plus un coût de montage, il suffit de modifier la méthode `calculerPrix()` de la classe `Ordinateur`.

Nous verrons par la suite qu'il est possible de faire bien mieux encore en se servant de ce qui est appelé l'héritage.


REMARQUE

la POO en deux mots

Ceci n'est qu'un aperçu de la programmation orientée objet (POO) ; un aperçu qui n'a d'ailleurs pas d'autre ambition que d'initier. Il existe de nombreux livres et sites Internet qui en parlent longuement, de manière générale ou associé à un langage en particulier.

Définir une classe

Il est d'usage de définir une seule classe par fichier, mais cela n'est pas obligatoire. Voici le squelette d'une classe :

```
<?php
class Ordinateur
{
    // on placera ici le constructeur, les attributs et les méthodes.
}
?>
```

Une classe, un fichier
Une classe par fichier est sans aucun doute la meilleure façon de s'y retrouver : le nom du fichier peut ainsi prendre le nom de la classe (éventuellement suffixé par "_class"), et il est alors très facile de retrouver la classe sur laquelle on veut travailler.

stdClass
Vous ne devez en aucun cas appeler une classe `stdClass`, ce nom étant réservé à PHP.

Les constructeurs

Un constructeur est une méthode (fonction) appelée lors de l'instanciation (la création d'un objet), qui sert généralement à initialiser des attributs. Le nom du constructeur doit être "`__construct`".

Compatibilité ascendante
Avec PHP 4, le nom du constructeur devait être identique au nom de la classe. Ne vous inquiétez pas ! Ce n'est pas parce que vous avez, dans vos tiroirs, des scripts écrits en objet pour PHP 4, que vous devez nécessairement vous empresser de les corriger pour les faire fonctionner avec PHP 5. En fait, si la méthode `__construct` n'est pas trouvée, l'interpréteur continuera à rechercher une méthode portant le même nom que la classe et s'en servira comme constructeur.

```
<?php
class Ordinateur
{
    // on placera ici les attributs.

    function __construct($boitier, $ecran, $clavier, $souris)
    {
        echo "Le constructeur a été appelé avec les paramètres:<br />";
        echo $boitier."<br />";
        echo $ecran."<br />";
        echo $clavier."<br/>";
        echo $souris."<br />";
    }

    // on placera ici les méthodes.
}
?>
```

Pour créer un objet, il suffit d'utiliser l'instruction `new` comme suit :

```
$monOrdinateur = new Ordinateur($monBoitier, $monEcran,
                                $monClavier, $maSouris);
```

L'exemple ici suppose que `$monBoitier`, `$monEcran`, `$monClavier` et `$maSouris` sont de type chaîne de caractères (ce qui n'est pas ce que l'on souhaite à terme).

A tout moment, il vous est possible de vérifier si un objet appartient à (ou dérive d'une) classe donnée en faisant appel à l'instruction `instanceOf`.

```
<?php
    // Nous supposons que la classe MaClasse a été définie précédemment
    $obj = new MaClasse();
    if ($obj instanceOf MaClasse) {
        echo "Oui, c'est bien le type d'objet que j'attendais";
    } else {
        echo "Mais c'est quoi cet objet ?";
    }
?>
```


Les attributs

Les attributs sont des variables ou des constantes associées à un objet. Ils sont précédés du mot-clé `public`, `protected` ou `private`.

```
<?php
class Ordinateur
{
    public $boitier;
    public $ecran;
    public $clavier;
    public $souris;

    function __construct($boitier, $ecran, $clavier, $souris)
    {
        $this->boitier = $boitier;
        $this->ecran   = $ecran;
        $this->clavier = $clavier;
        $this->souris  = $souris;
    }

    // on placera ici les méthodes.
}
?>
```


REMARQUE

var PHP4

Du temps de PHP4, les modificateurs de portée: `public`, `protected` et `private` n'existaient pas. Il fallait alors utiliser le mot clé `var` (qui reste toutefois compatible avec PHP 5 et qui se comporte comme `public`)

Dans cet exemple, le constructeur joue bien son rôle d'initialisation. `$this` est l'objet courant. `$this->boitier` fait donc appel à l'attribut `boitier` de l'instance "courante" de `Ordinateur`.

Les attributs ayant été déclarés `public`, ils sont accessibles depuis "tout endroit du code": à savoir au sein de la classe (via `$this->boitier`) tout comme en dehors de la classe par l'instruction :

```
$monBoitier = $monOrdinateur->boitier;
```

Et comme nous le verrons plus tard, un attribut `public` est également accessible depuis une classe dérivée (Cf. héritage).

Appel d'attributs

Une erreur courante consiste à écrire `$monOrdinateur->$boitier` au lieu de `$monOrdinateur->boitier`. Bien entendu, la première version ne peut fonctionner, car elle devient `$monOrdinateur->""` si `$boitier` n'est pas défini.

Les constantes

Il est possible de définir des constantes au sein d'une classe, pour cela, il suffit d'utiliser l'instruction `const`. Celles ci sont alors uniquement accessibles via un appel statique (`MonObjet::MA_CONSTANTE`).

Notez qu'une constante définie au niveau de la classe sera prioritaire sur une constante portant le même nom, mais définie en dehors de la classe via l'instruction `define`.

```
<?php
    class Divers
    {
        const MODE_RAPIDE = "rapide"
        const MODE_PRECIS = "precis";
    }

    echo Divers::MODE_RAPIDE."<br />";
?>
```

Valeur initiale des attributs

Comme pour les variables statiques des fonctions, il n'est pas possible d'initialiser les valeurs des variables en faisant appel à des opérateurs (ex: `public $val = 5 + 7;`) ou à des fonctions (ex: `public $val = sqrt(25);`).

attributs statiques

Au besoin, vous pourrez également utiliser le modificateur "`static`" qui permet de définir une variable partagée par toutes les instances de l'objet. Une variable statique s'adresse selon le schéma `MonObjet::$maVariable`.

```
<?php
    class Divers
    {
        static $compteur = 0;
        function incrementeCompteur()
        {
            self::$compteur++;
        }
        function afficheCompteur()
        {
            echo self::$compteur;
        }
    }

    $obj1 = new Divers();
    $obj1->incrementeCompteur();
    $obj2 = new Divers();
    $obj1->afficheCompteur();
    $obj2->afficheCompteur();
?>
```

affichera donc :

```
2
2
```

Ce qui montre bien que la variable statique est partagée par les deux instances.

On n'est jamais mieux servi que par soi-même
Dans le cas d'un attribut ou d'une méthode statique, pour faire référence à la classe courante vous devez utiliser l'instruction `self::`.

Les méthodes

Les méthodes sont des fonctions propres à une classe donnée qui permettent de manipuler ou accéder aux attributs d'un objet. Elles se déclarent comme des fonctions mais sont définies au sein d'une classe.

```
<?php
class Ordinateur
{
    public $boitier;
    public $ecran;
    public $clavier;
    public $souris;

    function __construct($boitier, $ecran, $clavier, $souris)
    {
        $this->boitier = $boitier;
```

```
        $this->ecran = $ecran;
        $this->clavier = $clavier;
        $this->souris = $souris;
    }

    // La fonction calculer_prix renvoie la somme des prix des composants
    function calculerPrix()
    {
        $boitier = $this->boitier;
        $ecran = $this->ecran;
        $clavier = $this->clavier;
        $souris = $this->souris;
        return $boitier->prix+$ecran->prix+$clavier->prix+$souris->prix;
    }
?>
```

L'appel à une méthode se fait selon le schéma suivant:

```
$monObject->maMethode();
```


REMARQUE

Noms de méthodes
Deux classes peuvent avoir les mêmes noms de méthodes, vu qu'elles sont précédées, lors de leur appel, d'un objet typé. Aucune confusion n'est alors possible.

Les remarques, présentées dans le chapitre sur les fonctions, concernant la portée des variables (globales, statiques ou locales) s'appliquent exactement de la même manière pour les méthodes.


ATTENTION

Noms de méthodes réservés
Afin de ne pas prendre le risque de rentrer en conflit avec des méthodes (appelées "fonctions magiques") utilisées pour le fonctionnement interne de PHP, vous ne devez pas faire précéder le nom de vos méthodes par "__".

Les méthodes statiques

Une méthode ne faisant pas appel à un attribut de l'objet (ou à une autre méthode non statique) est dit statique.

```
<?php
class Souris
{
    function afficher()
    {
        echo "Cet objet représente une souris d'ordinateur";
    }
}
?>
```

Pour appeler une telle méthode, il est inutile de créer un objet (par un appel `new`). Il est donc possible d'utiliser l'opérateur `::` (au lieu de `->`) . Ainsi, le script suivant aura pour effet d'afficher "Cet objet représente une souris d'ordinateur" sans pour autant avoir eu besoin de créer un objet *Souris*.

```
<?php
    Souris::afficher();
?>
```

Passage par référence et déréférencement

Passage par référence

Avec PHP 5, les objets sont systématiquement passés par référence (comme dans un langage tel que Java) et non plus par copie. Il n'est donc pas nécessaire de faire précéder le nom du paramètre d'un `&` comme ce fût le cas avec PHP 4.

Ainsi, l'exemple suivant :

Listing 3.36 : Exemple

```
<?php
    class MaClasse
    {
        var $msg;
        function MaClasse()
        {
            $this->msg = "Message par défaut";
        }
    }
    function modifieMessage($objet)
    {
       $objet->msg = "Message modifié par la fonction";
    }
    $obj = new MaClasse();
    modifieMessage($obj);
    echo $obj->msg;
}
```

retournera

```
Message modifié par la fonction
```

alors qu'avec PHP 4, il retournait

```
Message par défaut
```

puisque la modification portait sur une copie de l'objet.


REMARQUE

Objet retour
Bien que cela soit plus difficile à démontrer, c'est également vrai pour les objets retournés par les fonctions (par `return`). Dans ce cas, ce n'est pas une copie de l'objet créé dans la fonction qui est retourné, mais une référence sur l'objet.

En fait, ce sont toutes les manipulations d'objets qui se font par référence (comme c'est traditionnellement le cas avec un langage de programmation orienté objet) et non plus par copie.

Ainsi le code suivant :

```
<?php
    $obj = new MaClasse();
    $obj2 = $obj1;
    $obj2->msg = "Message attribué à \$obj2";
    echo $obj1->msg;
?>
```

retournera:

```
Message attribué à $obj2
```

Alors qu'avec PHP 4, il retournerait :

```
Message par défaut
```

Déréférencement

A supposer que l'une des méthodes de l'objet retourne un objet comme par exemple la méthode suivante:

```
function recupereEcran()
{
    return $ecran;
}
```

Il est désormais possible de récupérer en un seul appel la valeur d'un attribut de l'objet ainsi retourné en utilisant la syntaxe `$ordinateur->recupereEcran()->prix`. On parle alors de déréférencement.


REMARQUE

PHP 4 et le déréférencement
Notez bien que cela n'était pas possible avec PHP 4. Nous étions alors contraints de passer par une variable intermédiaire, selon le modèle suivant: `$ecran = $ordinateur->recupereEcran()`, puis `$ecran->prix`.
L'appel suivant était, en revanche, tout à fait valide `$ordinateur->ecran->prix`.

Typage

Sans que PHP ne soit devenu un langage fortement typé avec la venue de la version 5. Celle-ci permet toutefois de préciser le type (uniquement s'il s'agit d'un objet) des paramètres des méthodes. Comme le montre l'exemple suivant (la classe Boitier est décrite un peu plus loin):

```
function changerBoitier(Boitier $boitier)
{
    $this->boitier = $boitier;
}
```

En cas de passage de paramètre d'un mauvais type, l'erreur ne sera pas détectée lors de la compilation mais lors de l'exécution de la méthode. Il s'agit alors d'une erreur de niveau "fatal" et non une exception qui est levée.

Modificateurs de méthodes

Sur le même principe que pour les attributs, la déclaration des méthodes peut être affinée grâce aux modificateurs "`private`", "`protected`" ou par défaut "`public`" pour ce qui est de la portée et "`final`" pour ce qui est des droits en modification (voir héritage).

L'héritage

Si l'on reprend l'ensemble des classes (La classe `Ordinateur` ayant été précédemment définie) nous obtenons:

Classe Boitier

```
<?php
class Boitier
{
    public $modele;
    public $details;
    public $prix;

    // Constructeur
    function __construct($modele, $prix, $details)
    {
        $this->modele = $modele;
        $this->details = $details;
        $this->prix = $prix;
    }

    // Méthodes.
    function modifierModele($modele)
    {
        $this->modele = $modele;
    }

    function modifierDetails($details)
```

```
    {
        $this->details = $details;
    }

    function modifierPrix($prix)
    {
        $this->prix = $prix;
    }
?>
```

Classe Ecran

```
<?php
class Ecran
{
    public $modele;
    public $prix;

    // Constructeur
    function __construct($modele, $prix)
    {
        $this->modele = $modele;
        $this->prix = $prix;
    }

    // Méthodes.
    function modifierModele($modele)
    {
        $this->modele = $modele;
    }

    function modifierPrix($prix)
    {
        $this->prix = $prix;
    }
?>
```

Classe Clavier

```
<?php
class Clavier
{
    public $modele;
    public $prix;

    // Constructeur
    function __construct($modele, $prix)
    {
        $this->modele = $modele;
        $this->prix = $prix;
    }
```

```
    // Méthodes.
    function modifierModele($modele)
    {
        $this->modele = $modele;
    }

    function modifierPrix($prix)
    {
        $this->prix = $prix;
    }
?>
```


Classe Souris

```
<?php
class Souris
{
    public $modele;
    public $prix;

    // Constructeur
    function __construct($modele, $prix)
    {
        $this->modele = $modele;
        $this->prix = $prix;
    }

    // Méthodes.
    function modifierModele($modele)
    {
        $this->modele = $modele;
    }

    function modifierPrix($prix)
    {
        $this->prix = $prix;
    }
?>
```

Ces quatre dernières classes se ressemblant énormément, il serait préférable de créer une classe commune `Composant`. C'est l'objet de ce sous-chapitre.

La programmation objet permet de faire de l'héritage. On parle d'héritage lorsqu'une classe bénéficie (hérite) des propriétés (méthodes et attributs) d'une autre. La classe qui hérite des propriétés pourra implémenter d'autres propriétés ; on dit alors quelle "étend" la classe parente.

Dans notre exemple, nous allons créer une classe `Composant` dont hériteront les classes `Boitier`, `Clavier` et `Souris`.

Classe Composant

```
<?php
class Composant
{
    public $modele;
    public $prix;

    // Constructeur
    function __construct($modele, $prix)
    {
        $this->modele = $modele;
        $this->prix = $prix;
    }

    // Méthodes.
    function modifierModele($modele)
    {
        $this->modele = $modele;
    }

    function modifierPrix($prix)
    {
        $this->prix = $prix;
    }
?>
```

Maintenant, si les classes `Boitier`, `Clavier` et `Souris` héritent de la classe `Composant`, elles vont considérablement se simplifier. Pour cela, il suffit de préciser "`extends <classe parente>`" lors de la déclaration de la classe.

Classe Souris

```
<?php
class Souris extends Composant
{
}
?>
```

Cette déclaration suffit à définir la classe `Souris` comme fait précédemment. Cette classe vide peut sembler inutile, mais elle peut s'avérer très utile si l'on veut ajouter une information sur le nombre de boutons ou sur le fait qu'elle soit sans fil, par exemple (information qui n'aurait pas de sens pour les autres composants).

Nous verrons avec la classe `Boitier` comment ajouter des méthodes propres à une classe.

Classe Clavier

```
<?php
class Clavier extends Composant
{
```

```
}
?>
```

Ce fichier suffit à définir la classe `Clavier` comme fait précédemment. De même que pour la classe `Souris`, il pourrait être intéressant d'ajouter le nombre de touches ou le fait que ce soit un clavier avec ou sans fil.

Classe Boitier

```
<?php
class Boitier extends Composant
{
    private $details;

    // Constructeur
    function __construct($modele, $prix, $details)
    {
        new Composant($modele,$prix);
        $this->details = $details;
    }

    // Méthodes.
    function modifierDetails($details)
    {
        $this->details = $details;
    }
}
?>
```

Ainsi défini, la classe `Boitier` bénéficie des méthodes `modifierModele()` et `modifierPrix()` définies dans la classe `Composant`. Mais comme la classe que nous souhaitons obtenir est "plus riche" que la classe `Composant` dont elle hérite, il a fallu redéfinir le constructeur, et ajouter une méthode et un attribut.

La classe `Ordinateur` reste inchangée.

Dans le cas de l'héritage, la classe parente peut être référencée à l'aide du mot-clé `parent`. Ainsi, l'instruction `parent::modifierModele()` fera appel à la fonction `modifierModele()` de la classe parente même si celle-ci a été redéfinie au niveau de la classe fille.


REMARQUE

Héritage et appel du constructeur
Lorsque la classe qui hérite ne redéfinit pas le constructeur, alors c'est le constructeur de la classe parente qui est appelé.
Lorsque la classe qui hérite redéfinit le constructeur, alors c'est le constructeur de la classe fille qui est appelé, sans que le soit le constructeur de la classe parente (à moins d'un appel explicite `parent::__construct()`). Ce comportement est différent de celui de certains langages, et de Java en particulier.


REMARQUE

Héritage multiple
Il n'est pas certain que PHP intègre un jour l'héritage multiple mais comme le prouve Java, on peut très bien vivre sans (tout en s'évitant bien des soucis).

Réécriture de méthode

Comme nous l'avons vu avec l'exemple de la classe `Boitier`, une classe fille peut très bien ré-écrire une méthode de la classe mère. Dans ce cas, la signature de la méthode reste inchangé mais son comportement pourra être totalement (ou en partie modifié).

Si vous ne souhaitez pas que vous ou un quelconque utilisateur de votre classe ne fasse une grosse bourde en réécrivant maladroitement une des méthodes de la classe alors déclarez là `final` .

```
<?php
class ClassMereAvecUneMethodeCritique
{
    final function methodeCritique()
    {
        // C'est une methode critique a ne surtout pas
        // ré-écrire.
    }
}
?>
```

Portée des attributs et méthodes (public, protected, private)

Comme cela a été évoqué précédemment, les attributs et les méthodes peuvent être déclarés `public`", "`protected`" ou "`private`".

Un attribut ou une méthode déclaré `public` est accessible librement.

Un attribut ou une méthode déclaré `private` n'est accessible que depuis une méthode de la classe dans laquelle il a été défini.

Un attribut ou une méthode déclaré `protected` n'est accessible que depuis une méthode de la classe dans laquelle il a été défini ou d'une classe en héritant.

L'utilisation de ces modificateurs de portée n'est pas à négliger. Il peut être par exemple nécessaire de rendre des attributs `private` ou `protected` afin d'obliger l'utilisation d'une méthode pour affecter ces valeurs. Ces méthodes appelées accesseurs pourront au besoin pré-traiter la demande ou en contrôler la validité avant de modifier ou retourner la valeur de l'attribut.

```
<?php
class Composant
{
    private $modele;
    private $prix;

    // Constructeur
```

```
    function __construct($modele, $prix)
    {
        $this->modele = $modele;
        $this->prix = $prix;
    }

    // Méthodes.
    function modifierModele($modele)
    {
        if ($modele == "inconnu") $modele = NULL;
        $this->modele = $modele;
    }

    function modifierPrix($prix)
    {
        if ($prix < 0) $prix = 0;
        $this->prix = $prix;
    }
?>
```

Avec cette classe, l'appel `$monBoitier->prix = 10;` n'est plus possible. Il faut nécessairement passer par la méthode `modifierPrix()` comme suit `$monBoitier->modifierPrix(10);` ce qui permettra de vérifier que le prix indiqué n'est pas négatif.

Les interfaces

Lorsqu'elle est correctement maîtrisée, la programmation orientée objet permet de faire des choses vraiment formidables. Il est ainsi possible de créer des classes (et leurs méthodes) qui manipulent des objets de façon tout à fait générique sans en connaître leur implémentation exacte mais en s'appuyant simplement sur les méthodes qu'ils proposent. Dans ce cas, il faut toutefois s'assurer que les objets manipulés implémentent effectivement ces méthodes. Pour s'en assurer, il suffit alors de définir une interface décrivant l'ensemble des méthodes qui doivent être implémentées.

```
<?php
interface MonInterface {
    public function afficher();
}
?>
```

Nous avons ici décrit l'interface d'une classe devant implémenter une méthode `afficher()`. Les classes prétendant implémenter cette interface devront alors le préciser avec le mot clé `implements`.

```
<?php
class ClasseQuiImplemente implements MonInterface {
    function afficher()
    {
        echo "Moi quand j'affiche c'est à l'écran";
    }
```

```
}
?>
```

Une telle classe pourra alors être utilisée par la classe suivante:

```
<?php
class ClasseQuiFaitLeTraitement
{
    var $obj;
    function __construct($obj)
    {
        $this->obj = obj;
    }
    function traitement()
    {
        // Quelques lignes de traitement puis
        $this->obj->afficher();
    }
}

$objAffichage = new ClasseQuiImplemente();
$objTraitement = new ClasseQuiFaitLeTraitement($objAffichage)
$objTraitement->traitement();
?>
```


Les classes abstraites

Un peu dans le même esprit que les interfaces, il existe également les classes abstraites. Il s'agit alors de classes qui peuvent implémenter un certain nombre de méthodes et laisse le soin à d'autres d'implémenter un certain nombre d'autres méthodes alors déclarées comme `abstract` (abstraites). Une classe contenant des méthodes abstraites est elle-même abstraite et ne peut être instanciée. Une classe abstraite ne peut être que dérivée (par héritage).

L'exemple précédent donnerait alors

```
<?php
abstract class MaClasseAbstraite {
    abstract public function afficher();
    function autreMethode()
    {
        // Eventuellement une methode qui
        // a été implémentée ici
    }
}

class ClasseQuiImplemente extends MaClasseAbstraite {
    function afficher()
    {
        echo "Moi quand j'affiche c'est à l'écran";
    }
}
```

```
class ClasseQuiFaitLeTraitement
{
    var $obj;
    function __construct(MaClasseAbstraite $obj)
    {
        $this->obj = obj;
    }
    function traitement()
    {
        // Quelques lignes de traitement puis
        $this->obj->afficher();
    }
}

$objAffichage = new ClassQuiImplemente();
$objTraitement = new ClasseQuiFaitLeTraitement($objAffichage)
$objTraitement->traitement();
?>
```

Vous noterez que dans ce cas, il est possible de vérifier que le classe passée en paramètre du constructeur hérite bien de la classe abstraite (et donc implémente bien la méthode `afficher()`).

Les exceptions

L'ensemble des langages de programmation orientée objet offrent une alternative à l'habituel code d'erreur retourné par une fonction: à la place, lorsqu'une classe lève une erreur, elle interrompt son déroulement et redonne la main au programme appelant en émettant un objet appelé `Exception`. PHP n'échappe pas à cette règle.

La génération de ces "messages" est alors contrôlée et gérée dans un bloc `try`... `catch`..

Listing 3.37 : exception.php

```
<?php

class ObjetMathematique
{
    function division($a, $b)
    {
        if ($b == 0) throw new Exception("Division par zero.", 10);
        else return $a/$b;
    }
}

try  {
    $obj = new ObjetMathematique();
    echo $obj->division(4, 2)."<br />\n";
    echo $obj->division(4, 0)."<br />\n";
    echo $obj->division(8, 4)."<br />\n";
} catch (Exception $ex)  {
    echo "getMessage() ".$ex->getMessage()."<br />\n";
```

```
    echo "getCode() ".$ex->getCode()."<br />\n";
    echo "getFile() ".$ex->getFile()."<br />\n";
    echo "getLine() ".$ex->getLine()."<br />\n";
    echo "getTrace() ";
    var_dump($ex->getTrace());
    echo "<br />\n";
    echo "getTraceAsString() ".$ex->getTraceAsString()."<br />\n";
}
?>
```

Cela retournera alors :

```
2<br />
getMessage() Division par zero.<br />
getCode() 10<br />
getFile() /usr/local/apache/htdocs/regtest/poo/_exception.php<br />
getLine() 7<br />
getTrace() array(1) {
  [0]=>
  array(6) {
    ["file"]=>
    string(51) "/usr/local/apache/htdocs/regtest/poo/_exception.php"
    ["line"]=>
    int(15)
    ["function"]=>
    string(8) "division"
    ["class"]=>
    string(17) "ObjetMathematique"
    ["type"]=>
    string(2) "->"
    ["args"]=>
    array(2) {
      [0]=>
      int(4)
      [1]=>
      int(0)
    }
  }
}
<br />
getTraceAsString() #0 /usr/local/apache/htdocs/regtest/poo/_exception.php(15):
✂ ObjetMathematique->division(4, 0)
#1 {main}<br />
```

En effet, lorsque nous avons appelé la méthode division avec un diviseur égal à 0, la méthode a levé une exception grâce à l'instruction `throw`. Ceci a donc mis un terme à l'exécution de la méthode ainsi qu'à celle du bloc `try` du programme appelant, l'exécution se poursuivant par le bloc `catch` correspondant à l'objet exception généré. (Ici, nous n'avons qu'un bloc `catch`, mais il est possible d'en avoir autant qu'il y a d'exceptions différentes). Le reste du programme (à la suite du bloc `try...catch`) se déroule alors normalement.

Dans cet exemple, nous avons utilisé la classe prédéfinie `Exception`, nous aurions également pu définir notre propre classe d'exception à condition qu'elle hérite de `Exception`.

La classe `Exception` présente un constructeur permettant de préciser un message d'erreur ainsi qu'un code d'erreur.

Exception->__construct()

Instancie un objet `Exception`.

Syntaxe	`new Exception([string $message [, int $codeErreur]])`
`$message`	Message d'erreur.
`$codeErreur`	Code d'erreur.

Cette classe possède de nombreuses méthodes:

Exception->getMessage()

Retourne le message d'erreur de l'`Exception`.

Syntaxe	`string getMessage()`
`retour`	Message d'erreur.

Exception->getCode()

Retourne le code d'erreur de l'`Exception`.

Syntaxe	`int getCode()`
`retour`	Code d'erreur.

Exception->getFile()

Retourne le nom du fichier dans lequel l'`Exception` a été levée.

Syntaxe	`string getFile()`
`retour`	Nom du fichier.

Exception->getLine()

Retourne le numéro de la ligne du fichier où l'`Exception` a été levée.

Syntaxe	`int getLine()`
`retour`	Ligne dans le fichier.

Exception->getTrace()

Retourne le tableau des enchaînements des appels de fonctions ayant conduit à l'`Exception`.

Syntaxe	`array getTrace()`
`retour`	Tableau indexé contenant une entrée par fonction dans la pile d'appel. Chacune des valeurs de ce tableau est un tableau associatif contenant les clés: `file`, précisant le nom du fichier contenant l'appel; `.line`, précisant la ligne où a eu lieu l'appel, `function`, précisant le nom de la fonction (ou méthode) appelée, `class`, précisant le nom de la classe impliquée, `type`, précisant le type d'appel (ex: '->' pour un appel de méthode non statique) et enfin `args`, précisant les paramètres passés à la fonction.


REMARQUE

PHP 4 (ne) fait (pas) Exception
PHP 4 ne gérait pas les exceptions.

Les fonctions de manipulation des objets

PHP propose un ensemble de fonctions applicables à des objets. Il s'agit essentiellement de fonctions d'introspection: c'est à dire des fonctions retournant des informations sur un objet comme par exemple la liste des attributs et méthodes qu'il expose.

get_class()

Retourne le nom de la classe dont l'objet spécifié est une instance.

Syntaxe	`string get_class(object $objet)`
`$objet`	Objet dont vous souhaitez déterminer la classe.
`retour`	Nom de la classe ou `FALSE` en cas d'échec.

class_exists()

Vérifie si une classe existe ou non.

Syntaxe	`boolean class_exists(string $classe [, boolean $chargementAutomatique])`
`$classe`	Nom de la classe que vous souhaitez tester.
`$chargement Automatique`	`TRUE` (valeur par défaut) si vous souhaitez utiliser le chargement automatique de la classe (cf. `__autoload`), `FALSE` sinon.
`retour`	`TRUE` si la classe existe, `FALSE` sinon.

get_class_methods()

Retourne la liste des méthodes exposées par une classe ou un objet.

Syntaxe	`array get_class_methods(mixed $classe)`
`$classe`	Nom de la classe ou une instance (l'objet directement).
`retour`	Tableau indexé ayant pour valeurs les noms des méthodes ou `NULL` si la classe n'existe pas.

method_exists()

Vérifie si un objet expose une méthode donnée ou non.

Syntaxe	`boolean method_exists(object $objet, string $methode)`
`$objet`	Objet que vous souhaitez tester (contrairement à `get_class_method()` il n'est pas possible de spécifier le nom de la classe à la place).
`$classe`	Nom de la méthode.
`retour`	`TRUE` si la méthode existe, `FALSE` sinon.

get_class_vars()

Retourne la liste des attributs d'une classe et leur valeur (initiale).

Syntaxe	`array get_class_vars(string $classe)`
`$classe`	Nom de la classe (contrairement à `get_class_method()` il n'est pas possible de spécifier l'objet directement).
`retour`	Tableau indexé ayant pour clés les noms des attributs et pour valeur la valeur initiale des attributs ou `FALSE` si la classe n'existe pas.

get_object_vars()

Retourne la liste des attributs d'un objet et leur valeur.

Syntaxe	`array get_object_vars(object $objet)`
`$objet`	Objet dont vous souhaitez obtenir la liste de valeur des attributs.
`retour`	Tableau indexé ayant pour clés les noms des attributs et pour valeur la valeur des attributs ou `FALSE` en cas d'erreur.

D'autres fonctions sont plus spécifiquement liées à la notion d'héritage.

get_parent_class()

Retourne le nom de la classe mère de la classe testée.

Syntaxe	`string get_parent_class(mixed $classe)`
`$classe`	Nom de la classe ou instance (l'objet directement).
`retour`	Nom de la classe mère ou `FALSE` en cas d'erreur.

is_subclass_of()

Vérifie si un objet hérite d'une classe donnée ou non.

Syntaxe	`boolean is_subclass_of(object $objet, string $classe)`
`$objet`	Objet à tester.
`$classe`	Nom de la classe dont l'objet est susceptible d'hériter.
`retour`	`TRUE` si l'objet hérite bien de la classe indiquée, `FALSE` sinon.

Il est également possible de récupérer la liste de l'ensemble des classes et interfaces connues du script.

get_declared_classes()

Retourne la liste des classes connues (on y retrouve évidemment stdClass, Exception, etc.).

Syntaxe	`array get_declared_classes()`
`retour`	Tableau indexé ayant pour valeurs les noms des classes.

get_declared_interfaces()

Retourne la liste des interfaces connues.

Syntaxe	`array get_declared_interfaces()`
`retour`	Tableau indexé ayant pour valeurs les noms des interfaces.

Programmation avancée

Mémoriser des objets

Il est parfois intéressant de stocker un objet dans un fichier (une variable de session ou encore une base de données). Ce n'est pas, a priori, une tâche aisée, mais elle se révèle très simple s'il est possible de transformer un objet en chaîne de caractères et une chaîne de caractères en objet. C'est ce que l'on appelle la sérialisation et la désérialisation.

La sérialisation s'effectue à l'aide de la fonction `serialize()` ; l'opération inverse à l'aide de la fonction `unserialize()`.

Pour pouvoir relire cette chaîne de caractères et la transformer en objet, la classe doit être définie au moment de l'opération de désérialisation.

Dans l'exemple suivant, nous sauvegarderons un objet *souris*, puis nous le récupérerons dans un autre script en se servant d'un fichier texte pour mémoriser cet objet.

Listing 3.38 : sauvegarde.php

```
<?php
    // Inclusion du contenu du fichier Souris.php
    // contenant la définition de la classe
    include("Souris.php");

    $souris = new Souris("Souris Facile",9.98);

    $chaine_souris = serialize($souris);

    $fichier = fopen("sauvegarde", "w"); // Le fichier sauvegarde
                                         //   est ouvert en écriture
```

```
    fputs($fichier, $chaine_souris);     // Le contenu de la chaîne de
                                          //   caractères est écrit
    fclose($fichier);                    // Le fichier est refermé
?>
```

Listing 3.39 : lecture.php

```
<?php
  // Il est indispensable d'inclure la définition de la classe
  // Souris pour que unserialize() fonctionne correctement
  include("Souris.php");

  $souris = implode("", @file("sauvegarde")); //la chaîne stockée
                                              // est récupérée

  unserialize($souris);  // la chaîne est transformée en son objet associé

  // $souris se traite désormais comme un objet.
  $souris->modifier_prix(9.99);
?>
```



REMARQUE

Les sessions

Dans le cas des sessions, si vous utilisez la fonction `session_register()` sur un objet, celui-ci est automatiquement sérialisé. Il faut donc faire bien attention à inclure la définition de la classe dans chacun des fichiers accessibles dès le moment où l'objet est enregistré en session. Sinon celui-ci se transforme en objet de classe `stdClass` et n'a aucune utilité, car, alors, aucune des méthodes n'est accessible.

__sleep

La fonction `__sleep()` est appelée avant toute sérialisation. Cette fonction renvoie un tableau des valeurs à mémoriser. Elle permet de n'indiquer que les valeurs utiles à stocker et de fermer les connexions aux bases de données par exemple.

__wakeup

La fonction `unserialize()` fait appel à cette méthode au moment de recréer l'objet. Cela peut permettre de rétablir des connexions aux bases de données par exemple.

```
<?php
class Test
{
    var $attribut1;
    var $attribut2;
    var $attribut3;

    function Test($attribut1, $attribut2)
    {
```

```
        $this->attribut1=$attribut1;
        $this->attribut2=$attribut2;
        $this->attribut3=$attribut1+$attribut2;
    }

    function __sleep() {
        return array('attribut1', 'attribut2');
    }

    function __wakeup() {
        $this->attribut3 = $this->attribut1 + $this->attribut2;
    }

}
?>
```

En faisant ceci, on économise en mémoire, car `$attribut3`, qui peut être calculé à partir de `$attribut1` et `$attribut2`, n'est pas stocké (en effet, la fonction `__sleep` ordonne de ne stocker que les attributs `attribut1` et `attribut2`). Il est cependant recalculé au moment de reconstituer l'objet.

Destruction des objets

Lorsqu'un objet n'est plus utilisé (ou tout simplement lorsque la fin du script arrive), les objets sont détruits. Jusqu'alors, il n'était pas possible de demander à PHP d'exécuter certaines fonctions (ex. : fermeture d'un fichier, déconnexion d'une base de données, etc.) avant de détruire définitivement l'objet. C'est désormais possible en écrivant une méthode `__destruct()` contenant le code à exécuter avant la destruction de l'objet.

```
<?php
    class ObjetMortBruyante
    {
        var $msg;
        function __construct()
        {
            $this->msg = "Je suis un objet qui ne sait pas mourir en silence";
        }
        function __destruct()
        {
             echo "Arrgggg.... je meurs";
        }
    }
    $obj = new ObjetMortBrutante();
    // la fin du script entraine la destruction de l'objet
?>
```

Ce script retournera donc :

```
Arrgggg.... je meurs
```

indiquant bien que la méthode `__destruct()` a été appelée.


REMARQUE

__destruct() et l'héritage

En cas d'héritage, tout comme pour le constructeur, le destructeur de la classe parente n'est pas appelé automatiquement. Si vous souhaitez appeler ce dernier vous devrez inclure un appel `parent::__destruct()`.

Clonage

Avec l'arrivée de PHP 5 et de son moteur Zend2, il est désormais possible de préciser comme doit se faire la copie d'un objet en créant une méthode `__clone()`.

Il est alors possible par exemple de réinitialiser certaines valeurs (les caractéristiques propres à la copie) tout en demandant la copie de certaines autres (les caractéristiques générales de l'objet copié). Pour accéder aux données de l'objet copié, vous devrez faire appel à `$that` (qui joue le rôle du `$this` mais pour l'objet copié).


REMARQUE

This or That ?

Nous devons avouer notre perplexité sur ce point. Il se trouve que dans les premières documentations et premières implémentations de PHP 5. Il fallait effectivement utiliser `$that`. *Cependant, l'exemple suivant qui fonctionnait tel que nous l'avions imaginé précédemment ne fonctionne pas exactement pareil avec PHP 5.0.1.*

Ainsi l'exemple suivant :

```
<?php
    class ObjetMutant()
    {
        var $msg, $msg2;
        function ObjetMutant()
        {
            $this->msg   = "Je suis un mutant";
            $this->msg2 = "et je reste un mutant.";
        }
        function __clone()
        {
             $this->msg = "J'ai muté";
             $this->msg2 = $that->msg2;
        }
    }
    $obj = new ObjetMutant();
    $obj2 = clone $obj;
    echo $obj->msg." ".$obj->msg2."<br />";
    echo $obj2->msg." ".$obj2->msg2."<br />";
?>
```

retournera donc (ou tout du moins retournait) :

```
Je suis mutant et je reste un mutant.
J'ai muté et je reste un mutant.
```

Il retourne désormais

```
Je suis mutant et je reste un mutant.
J'ai muté
```

A moins de remplacer le `$that` par un `$this`.

Quoiqu'il en soit, une chose est sure, à l'appel de l'instruction `clone`, la méthode `__clone()` a bien été exécutée. Notez, que la méthode `__clone()` ne peut être appelée directement.

Nouvelles méthodes "internes"

PHP 5 utilise de nouvelles méthodes internes.

En particulier lors de la récupération du contenu d'un attribut selon le schéma `$objet->attribut`, PHP retournera la valeur de l'attribut s'il existe, sinon il appellera la méthode `__get()` avec pour paramètre le nom de l'attribut. Méthode qu'il est possible de redéfinir.

```
<?php
class MaClasse
{
    var $attr1;
    function __contruct()
    {
        $attr1 = "demo";
    }
    function __get($nom)
    {
        return "Désolé, $nom n'est pas un attribut connu";
    }
}
$obj = new MaClasse();
echo $obj->attr1;
echo $obj->attr2;
?>
```

retournera donc:

demo

Désolé, attr2 n'est pas un attribut connu

Il en est de même avec l'affectation de valeur à une variable et la méthode interne `__set()` ainsi que pour l'appel de méthode et la fonction `__call()`. Dans ce dernier cas, la méthode reçoit deux paramètres, le premier étant le nom de la méthode appelée et le second étant un tableau indexé des paramètres de l'appel.

autres fonctions

Il est prévu de pouvoir définir son propre traitement lorsqu'il est fait référence à une classe qui n'existe pas. Pour cela, il suffit de redéfinir la fonction `__autoload()` qui accepte pour unique paramètre le nom de la classe appelée. Notez toutefois qu'à l'issu de l'appel à cette fonction, l'instanciation d'un objet de la classe invoquée devra être possible sinon le script s'arrêtera avec

un message d'erreur. En fait, cette fonction à pour objectif principal de vous permettre d'inclure au vol le fichier contenant la déclaration de la classe.

Les tableaux en POO

PHP propose un objet appelé `ArrayObject` chargé de manipuler un tableau comme n'importe quel objet.

ArrayObject->__construct()

Instancie un objet `ArrayObject`.

Syntaxe	`new ArrayObject([mixed $tableau])`
`$tableau`	L'objet `ArrayObject` peut être initialisé aussi bien avec un autre objet `ArrayObject` qu'avec un tableau "classique".

S'il n'est pas pré-rempli lors de l'appel du constructeur, l'objet peut être alimenté par des appels successif à la méthode `append()`.

ArrayObject->append()

Ajoute un élément à un objet `ArrayObject`.

Syntaxe	`void append(mixed $valeur)`
`$valeur`	Elément à ajouter à l'objet `ArrayObject`.

Cette méthode présente toutefois l'inconvénient majeur de ne pas permettre de préciser la clé associée à la valeur. Pour cela, nous privilégierons la méthode `offsetSet()`.

ArrayObject->offsetSet()

Ajoute un élément à un objet `ArrayObject` en précisant la clé associée.

Syntaxe	`void offsetSet(mixed $cle, mixed $valeur)`
`$cle`	Clé associée à l'élément à ajouter à l'objet `ArrayObject`.
`$valeur`	Elément à ajouter à l'objet `ArrayObject`.

A l'inverse, il est possible de récupérer la valeur associée à une clé par

ArrayObject->offsetGet()

Retourne l'élément d'un objet `ArrayObject` associé à une clé donnée.

Syntaxe	`mixed offsetGet(mixed $cle)`
`$cle`	Clé associée à l'élément à récupérer.
`retour`	Elément associé à la clé dans l'objet `ArrayObject`.

Pour supprimer un élément du tableau on fera appel à

ArrayObject->offsetUnset()

Supprime l'élément d'un objet `ArrayObject` associé à une clé donnée.

Syntaxe	`void offsetUnset(mixed $cle)`
`$cle`	Clé associée à l'élément à supprimer.

Il est également possible d'en vérifier l'existence

ArrayObject->offsetExists()

Teste la présence d'un élément d'un objet `ArrayObject` associé à une clé donnée.

Syntaxe	`boolean offsetExists(mixed $cle)`
`$cle`	Clé à rechercher.
`retour`	`TRUE`, si un des éléments du tableau est associé à la clé, `FALSE` sinon.

Il est évidemment, à tout moment possible de vérifier la taille du tableau

ArrayObject->count()

Retourne le nombre d'élément contenu dans un objet `ArrayObject`.

Syntaxe	`int count()`
`retour`	Nombre d'éléments dans le tableau.

Les itérateurs de tableau

Afin de parcourir le contenu d'un tableau, PHP met à disposition un objet `ArrayIterator`. Celui ci peut être obtenu en appelant la méthode `getIterator()` d'un `ArrayObject`.

ArrayObject->getIterator()

Retourne un itérateur sur un objet `ArrayObject`.

Syntaxe	`ArrayIterator getIterator()`
`retour`	Itérateur sur l'`ArrayObject`.

L'objet `ArrayIterator` agit comme un pointeur sur un élément du tableau. Pointeur qui permet de progresser du premier élément au dernier.

ArrayIterator->valid()

Vérifie que l'itérateur pointe toujours sur un élément de l'objet `ArrayObject`.

Syntaxe	`boolean valid()`
`retour`	`TRUE` si l'itérateur pointe sur un élément du tableau, `FALSE` sinon.

ArrayIterator->current()

Retourne l'élément "courant" du tableau (celui sur lequel l'itérateur pointe).

Syntaxe	`mixed current()`
`retour`	Valeur de l'élement du tableau actuellement pointé.

ArrayIterator->key()

Retourne la clé de l'élément "courant" du tableau (celui sur lequel l'itérateur pointe).

Syntaxe	`mixed key()`
`retour`	Clé de l'élement du tableau actuellement pointé.

ArrayIterator->next()

Déplace l'itérateur sur l'élément suivant du tableau.

Syntaxe `void next()`

Ainsi typiquement le parcours d'un tableau s'effectue comme ceci:

```
<?php
$tableau["cle1"] = "element1";
$tableau["cle2"] = "element2";
$tableau["cle3"] = "element3";
$monArrayObject = new ArrayObject($tableau);

echo "Ce tableau contient ".$monArrayObject->count()."  elements.<br />";

echo "J'utilise un iterateur pour le parcourir:<br />";
$iterateur = $monArrayObject->getIterator();
while ($iterateur->valid()) {
    echo "cle = ". $iterateur->key() .
         " valeur = ". $iterateur->current() .
         "<br />";
    $iterateur->next();
}
?>
```

Ceci dit l'itérateur peut être utilisé pour naviguer librement au sein du tableau.

ArrayIterator->seek()

Déplace le pointeur de l'itérateur sur un élément quelconque du tableau.

Syntaxe	`void seek(int $position)`
`$position`	Position dans le tableau (0 = premier élément) où doit être déplacé le pointeur de l'itérateur.

ArrayIterator->rewind()

Déplace le pointeur de l'itérateur sur le premier élément du tableau. Equivalent de `seek(0)`.

Syntaxe `void rewind()`

Listing 3.40 : arrayObject.php

```
<?php
$tableau[] = "element1";
$tableau[] = "element2";
$tableau[] = "element3";
$monArrayObject = new ArrayObject($tableau);

$monAutreArrayObject = new ArrayObject($monArrayObject);
echo "Ce tableau contient ".$monArrayObject->count()."  elements.<br />";

echo "J'utilise un iterateur pour le parcourir:<br />";
$iterateur = $monAutreArrayObject->getIterator();
echo "Je saute directement au 2eme element:<br />";
$iterateur->seek(1);
echo $iterateur->current()."<br />";

echo "Je reviens au premier:<br />";
$iterateur->rewind();
echo $iterateur->current()."<br />";
?>
```

Chapitre 4

Les en-têtes HTTP

L'utilisation explicite ou implicite des en-têtes va vous permettre de rediriger un client vers un autre site, et d'utiliser des cookies ou des variables de session afin de stocker (temporairement) des informations. Mais, avant d'aborder ces chapitres, il est important de bien comprendre le fonctionnement du dialogue client-serveur avec HTTP.

4.1. Principe général

Historique
Le protocole HTTP (HyperText Transfer Protocol), mis en œuvre lors de l'échange d'informations entre le client et le serveur, est une norme créée en 1990 par Tim Berners-Lee (qui est à présent président du Consortium du World Wide Web appelé plus communément W3C). Ce protocole est basé sur un système de communication par requêtes/réponses.

Ainsi, le client se connecte sur le serveur, sur un port spécifique (généralement le port 80), pour envoyer un message (que l'on appelle requête). Ce dernier doit respecter un schéma précis régi par une norme : ce doit être une méthode (de type `GET`, `POST`, `HEAD`, `PUT`, `DEL` ou `TRACE`) suivie d'une URI (adresse Internet). Au besoin, le client complète la requête à l'aide d'un message de type MIME (Multipurpose Internet Mail Extension). Il indique ainsi les informations particulières dont il a besoin, décline son identité (signature du client) et indique la version du protocole utilisé.

Le serveur, quant à lui, renvoie ensuite une réponse comprenant un en-tête HTTP comportant diverses informations sur le serveur et sur comment le client doit analyser le résultat, suivi du corps du document.

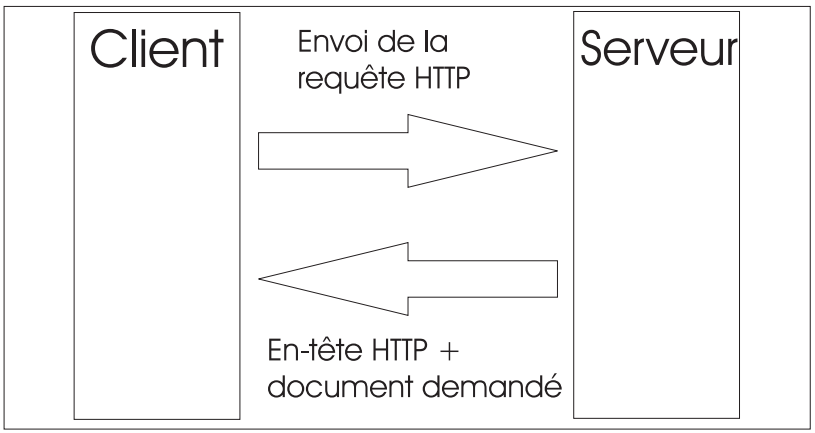


Figure 4.1 : *On peut schématiser une requête HTTP comme ceci*

Le client envoie donc un en-tête HTTP comprenant dans l'ordre :

- Une ligne de requête : elle comprend le type de document demandé ainsi que la méthode et la version du protocole qu'il lui est demandé d'utiliser. Cette ligne peut être schématisée comme ceci : `<METHODE> <URL> <VERSION>`.

Tableau 4.1 : Les différentes méthodes pouvant être utilisées

Méthode	Signification	Version
POST	La méthode POST est utilisée pour une soumission de données vers une URI cible.	HTTP/1.0
GET	La méthode GET est utilisée pour récupérer un contenu identifié par l'URI spécifiée.	HTTP/1.0
HEAD	La méthode HEAD est identique à la méthode GET sauf qu'elle ne réclame que l'en-tête du document spécifié. Le document lui-même n'est ainsi pas renvoyé.	HTTP/1.0
PUT	Permet le dépôt d'un fichier donné directement sur le serveur distant.	HTTP/1.1
DELETE	Commande demandant au serveur distant de détruire un document indiqué à l'URI spécifiée.	HTTP/1.1
TRACE	La méthode TRACE permet au client de voir ce qui est reçu par le serveur. Son utilisation est utile dans les phases de test ou de diagnostic.	HTTP/1.1
CONNECT	Cette méthode est réservée pour une utilisation par un proxy pouvant dynamiquement être basculé vers un tunnel (ex. : un tunnel SSL).	HTTP/1.1

- Les en-têtes de la requête (autant de lignes que nécessaire) : ces lignes facultatives (hormis l'information Host dans le cas du protocole HTTP/1.1) permettent de donner des informations complémentaires à la requête, comme l'origine du client et l'identification (ou signature) de ce client (nom du navigateur, système d'exploitation, etc.). Ces informations (une par ligne) sont données comme un identifiant suivi immédiatement de deux points ':', une espace et de sa valeur : <IDENTIFIANT>: <valeur>.

Tableau 4.2 : Exemple d'en-têtes pouvant être utilisés

Identifiant (en-tête)	Signification
Accept	Type de contenu que le navigateur est susceptible d'accepter (ex. : text/html, audio/x-aiff, image/jpeg, etc.).
Accept-Charset	Le type de caractère que le navigateur attend en réponse (ex : ISO-8859-1).
Date	Date et heure du client (ex. : Sat, 27 Apr 2002 15:59:53 GMT)
From	Spécifie une adresse e-mail pour le client.
Host	Nom de la machine cliente.
Referer	Adresse de la page depuis laquelle la requête a été effectuée.
User-Agent	Chaîne d'identification du client (ex. : Mozilla/5.001 [windows; U; NT4.0; en-us] Gecko/25250101 pour un navigateur Mozilla sur un système Windows NT4.0).


RENVOI

Vous pouvez vous reporter à l'annexe "En-têtes et variables externes" pour une liste plus complète d'en-têtes.

- Le corps de la requête : ces lignes sont utilisées dans le cas d'une méthode de type `POST` pour l'envoi de données (comme les paramètres d'un formulaire).

Comme aucune explication ne vaut un bon exemple, ouvrez un client telnet en ligne de commande,

```
telnet www.linux.org 80
```

et tapez la commande suivante :

```
GET / HTTP/1.0
```

Pour valider votre requête, finissez en tapant deux fois sur la touche Entrée.


REMARQUE

Le client telnet de Windows
Notez que le client telnet de Windows ne permet pas de visualiser ce que vous rentrez depuis le clavier. Alors, ne vous trompez pas dans la commande, et respectez bien les majuscules et minuscules.

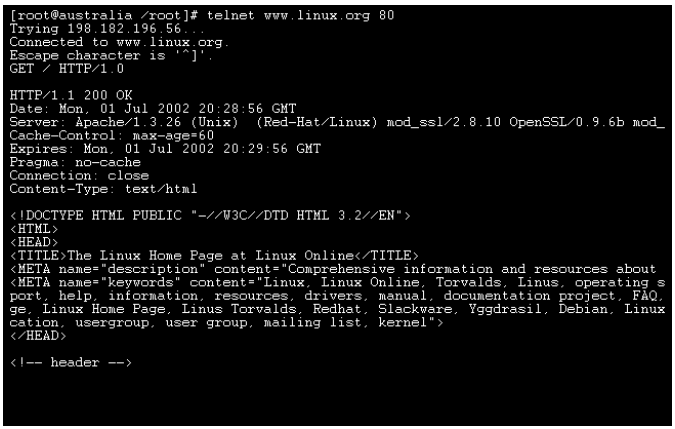


Figure 4.2 :
Voici une page telle qu'elle est envoyée par un serveur web

Vous voyez apparaître la réponse du serveur (ici, le code source de la page). Ce qui nous intéresse dans le cas présent, ce sont les premières lignes de ce document, à savoir l'en-tête de la page. Si vous voulez ne voir que cet en-tête, rentrez la commande `HEAD / HTTP/1.0`. Ainsi seul l'entête renvoyé par le serveur s'affichera dans le client telnet.

```
HTTP/1.1 200 OK
Date: Sat, 27 Apr 2002 15:59:53 GMT
Server: Apache/1.3.24 (Unix)  (Red-Hat/Linux) mod_ssl/2.8.8 OpenSSL/0.9.6b
✂ mod_perl/1.26
Cache-Control: max-age=60
Expires: Sat, 27 Apr 2002 16:00:53 GMT
```

```
Pragma: no-cache
Connection: close
Content-Type: text/html

<HTML>
...
```

Vous remarquez qu'il y a une ligne vide entre l'en-tête et le corps du document. C'est ce saut de ligne qui signale la fin de l'en-tête HTTP et indique que ce qui suit est le corps du document. Ainsi, une fois envoyé, l'en-tête ne peut plus être modifié. Vous verrez dans ce chapitre que le langage PHP permet des modifications de l'en-tête, mais cause une erreur si vous essayez de le modifier après l'envoi du début du document.

Vous pouvez exécuter à présent, toujours à l'aide de votre client telnet, la commande suivante :

```
telnet www.linux.org 80
GET / HTTP/1.1
host:www.linux.org
```

Comme indiqué précédemment, le protocole `HTTP/1.1` réclame une information supplémentaire `host`. En effet, cette dernière mouture de HTTP a permis l'adoption des hôtes virtuels, c'est-à-dire que plusieurs noms de domaines peuvent se partager une même adresse IP. D'autres avantages du protocole 1.1 sur le 1.0 sont répertoriés dans la RFC2616.

Les RFC

Pour toutes les informations complémentaires sur les protocoles `HTTP/1.0` et `HTTP/1.1`, consultez les RFC (Request For Comment). Ces documents ont la particularité de représenter une notice pour une documentation générale, un standard ou la description d'un protocole. Il est à noter que l'écriture même d'une RFC est définie dans une autre RFC (la RFC1543).

Pour HTTP/1.0, la RFC1945 est disponible à l'adresse **http://abcdrfc.free.fr/rfc-vf/rfc1945.html** *(traduction française).*

Pour HTTP/1.1, la RFC2616 est disponible (en anglais) à l'adresse **http://www.w3.org/Protocols/rfc2616/rfc2616.html**.

Ces documentations très complètes vous apprendront tout ce qu'il faut savoir pour effectuer un échange de données entre un navigateur Internet et un serveur web.

Voyons à présent ce que signifie cette réponse (i.e. comment le navigateur interprète ce message).

- La première ligne indique le code de retour. Ici le code 200 signifie que tout s'est bien passé. Tous les internautes connaissent au moins un autre code : le tristement célèbre 404, qui comme vous le savez, signifie que le serveur n'a pas trouvé la page demandée.

Tableau 4.3 : Les différentes catégories de code que peut renvoyer un serveur web et leurs significations

Intervalle de code	Signification
100 – 199	Les informations sont retournées.
200 – 299	La requête a été traitée avec succès.
300 – 399	Demande de redirection.
400 – 499	La requête est incomplète.
500 – 599	Indique une erreur du serveur HTTP.

Vous pouvez vous reporter à l'annexe sur les "Codes d'erreur HTTP" pour une liste plus détaillée.

Puis vient une série de valeurs précédées par un nom de paramètre. Avant de poursuivre, voici quelques paramètres d'en-tête possibles. Cette liste est non exhaustive, mais comprend les paramètres les plus courants qu'un serveur peut renvoyer.

Tableau 4.4 : Quelques-uns des messages que peut renvoyer un serveur dans son en-tête

Identifiant d'en-tête	Description
`Content-Encoding`	C'est le type de codage du document renvoyé (ex. : `compress`, `x-gzip`, `x-zip`, etc.).
`Content-Language`	C'est le langage utilisé dans le corps de la réponse ("fr" pour français, "en" pour anglais, "it" pour italien, etc.).
`Content-Length`	C'est la longueur de la réponse. Cette valeur est donnée en octets. Cette donnée permet au client d'être certain qu'il a bien reçu la totalité du document.
`Content-Type`	C'est le type MIME du document (ex. : `text/html`, `image/gif`, `application/postscript`, `text/plain`, `audio/basic`, `video/mpeg`, etc.).
`Date`	Date du serveur au début du transfert des données.
`Expires`	Date limite de validité du document (particulièrement utile dans le cas de l'utilisation d'un cache).
`Location`	Demande la redirection vers une nouvelle URL.
`Server`	Nom ou signature du serveur qui renvoie les informations.

Vous pouvez vous reporter à l'annexe "En-têtes et variables externes" pour un liste plus complète d'en-têtes.

Nous pouvons maintenant revenir à notre exemple et analyser sa réponse. Cet en-tête nous apprend donc plusieurs choses :

- La deuxième ligne de notre exemple indique simplement la date et l'heure du serveur web.
- La ligne suivante est la signature du serveur web. Dans notre exemple, le serveur hébergeant le site de **Linux.org** nous signale qu'il tourne sous le système d'exploitation Linux (est-ce étonnant ?) avec le serveur Apache et que celui-ci est compilé avec les modules SSL et Perl.
- La ligne `Cache-control` n'est valide que pour la version 1.1 du protocole HTTP. Cette ligne indique comment un client, et en particulier un proxy cache, doit traiter cette page (si le proxy doit mettre en cache la page, la relire à chaque fois, et combien de temps le cache de la page est valide. Ici, la page peut être archivée pendant 60 secondes au plus.)
- Pour les navigateurs utilisant le protocole HTTP/1.0, le serveur renvoie deux lignes à la suite qui permettent de définir la gestion du cache et la durée de celle-ci.
- Enfin, la ligne `Content-Type` indique à votre client web le type MIME de ce qui lui est envoyé. Cela peut être du texte, de l'HTML, une image, un fichier son, etc.


RENVOI

Vous pouvez consulter l'annexe sur "Les types MIME" pour une liste plus complète des types.

4.2. Gestion personnalisée de l'en-tête HTTP

D'une manière générale, avec PHP, il est possible de préciser la valeur des paramètres spécifiés dans l'en-tête, grâce à la fonction `header()`.

header()

Spécifie un élément de l'en-tête de la réponse HTTP.

Syntaxe	`void header(string $ligneEntete [, boolean $remplace [, int $reponseHTTP]])`
`$ligneEntete`	Information à ajouter à l'en-tête.
`$remplace`	`TRUE` si cette ligne doit remplacer la valeur donnée précédemment au même paramètre d'en-tête, `FALSE` (par défaut) si elle doit être ajoutée.
`$reponseHTTP`	Permet de forcer la reponse HTTP (par exemple 404 pour simuler une page manquante). Cette option est apparue avec la version 4.3.0 de PHP.


ATTENTION

Headers already sent...
Comme cela a déjà été évoqué, dès que des informations (corps du document) sont envoyées au navigateur, l'en-tête est envoyé au préalable. Par conséquent, il n'est plus question, à ce moment-là, de demander des modifications de cet en-tête.


ATTENTION

Concrètement, vous ne devez absolument pas "afficher" le moindre message avant un appel à `header()`*. Donc : pas de* `echo()`*,* `print()`*, etc., pas de message d'erreur et aucun code HTML ou texte quelconque, en dehors des balises* `<?php ... ?>`*. Faites également attention à ne pas laisser traîner d'espaces ou de lignes blanches avant ou après les balises* `<?php ... ?>` *des fichiers inclus avant l'appel à* `header()`*.*

Si besoin, vous disposez d'une fonction permettant de déterminer si l'en-tête a déjà été envoyé au client.

headers_sent()

Teste si l'en-tête HTTP a déjà été envoyé au client.

Syntaxe	`boolean headers_sent([string &$fichier [, int &$ligne]])`
`$fichier`	Si cette option est passee et qu'il a déjà été écrit sur la sortie, alors $fichier contiendra le nom du fichier dans lequel l'écriture a commencée (option disponible depuis PHP 4.3.0).
`$ligne`	Si cette option est passee et qu'il a déjà été écrit sur la sortie, alors $ligne contiendra le numero de la ligne dans lequel l'écriture a commencée (option disponible depuis PHP 4.3.0).
`retour`	`TRUE` si l'en-tête a déjà été émis, `FALSE` sinon.

Voici un court exemple de cette fonction :

```
<?php
  // rien ne va sur la sortie
  for ($i=0; $i<10; $i++) {
  // ici encore, rien n'est "ecrit"
  }
  $fichier="none";
  $ligne=0;

  echo "Par contre la l'entete est envoye a cause de echo()\n";

  if (headers_sent(&$fichier, &$ligne)==true)
    echo "L'entete a ete envoye par $fichier a la ligne $ligne\n";
  else
    echo "L'entete n'a pas ete envoye\n";
?>
```

Dont le résultat serait le suivant :

```
Par contre la l'entete est envoye a cause de echo()
L'entete a ete envoye par c:\program files\apache
✂ group\apache\htdocs\tests\headers_sent.php a la ligne 9
```

Les exemples d'applications sont nombreux ; nous en détaillerons trois dans ce chapitre.

Redirection

Il n'est pas rare d'avoir à rediriger le visiteur vers telle ou telle page en fonction de certains paramètres. Par exemple, si le visiteur est un homme, il peut être intéressant de l'envoyer vers le rayon homme d'un magasin de vêtements.

L'utilisation des en-têtes permet, entre autres, ce type de manipulation. Pour cela, il suffit d'utiliser l'en-tête `Location` avec, pour paramètre, l'adresse vers laquelle rediriger le navigateur.

Voici la syntaxe de la réorientation vers **http://www.php.net**:

```
header("Location: http://www.php.net");
```

La version 1.1 du protocole HTTP nécessite, comme paramètre, un chemin absolu. Si vous souhaitez faire une redirection au sein même de votre serveur vers un chemin relatif à la position du script appelé et être conforme à cette norme, voici un moyen de transformer une adresse relative en chemin absolu à l'aide du langage PHP.

```
header("Location: http://".$_SERVER['SERVER_NAME']
       ."/".dirname($_SERVER['PHP_SELF'])
       ."/".$chemin_relatif);
```

PHP se charge non seulement d'envoyer l'en-tête au navigateur, mais il lui retourne également le code 302 correspondant à `REDIRECT`.

Il est également possible de retourner directement un code au navigateur. Si vous voulez renvoyer le code 404 correspondant au code d'erreur d'un fichier inexistant, il suffit d'écrire :

```
header("HTTP/1.0 404 Not Found");
```

Voici un exemple de script redirigeant le navigateur du client en fonction du sexe de la personne ; on imagine que la variable `$sexe` a été indiquée précédemment.

```
<?php
if ($sexe == "homme") {
    header("Location: rayon_homme.html");
} else {
    header("Location: rayon_femme.html");
}
?>
```

Déclaration du type MIME

PHP est principalement utilisé pour générer du code HTML mais, comme nous le verrons par la suite, il permet également de générer toutes sortes de documents et notamment des images.

Or, par défaut, la configuration de PHP veut que le serveur déclare que le document émis est un document de type `text/html`. Dans tous les cas où le document émis n'est pas une page HTML, il convient donc d'en préciser le type, comme dans l'exemple suivant (s'il s'agit d'une image gif) :

```
header("Content-type: image/gif");
```

Vous pouvez vous reporter à l'annexe sur "Les types MIME" pour avoir une liste plus complète des valeurs possibles.

Gestion des caches (des navigateurs)

Généralement, les navigateurs (et éventuellement les systèmes intermédiaires comme les proxys) utilisent des caches pour stocker localement des documents (HTML, images, etc..) récupérés afin de ne pas avoir à les redemander aux serveurs lorsque ceux-ci sont rappelés (ex. : un logo que l'on trouve sur toutes les pages).

Si cela accélère grandement l'affichage des pages d'un site, il arrive qu'en certaines circonstances cela devienne un problème.

Le cas de figure le plus flagrant est certainement celui où le document est conservé une semaine dans le cache du navigateur, alors que vous mettez à jour la page quotidiennement. En fait, les navigateurs sont généralement configurés pour vérifier une fois par jour si le document demandé diffère de celui du cache. De ce fait, quelle que soit votre façon de procéder en tant que concepteur de sites web, vous êtes relativement à l'abri d'une plainte de ce côté-là. Le problème est plus "grave" si certains de vos scripts retournent un résultat différent en fonction de variables de sessions ou de données externes (base de données) qui varient d'un instant à l'autre, car l'appel au script, lui, ne varie pas (pas de paramètre `GET` différent d'un appel à l'autre et pas de paramètre `POST`). Le nom et les paramètres du script ne variant pas, le navigateur ira systématiquement prendre la version disponible dans le cache au lieu d'aller chercher sa nouvelle variante.

Dans ce cas, il est donc préférable de demander au navigateur (et autres proxys) de ne pas garder de version du document dans son cache, grâce aux appels suivants :

```
header("Cache-Control: no-cache");
header("Pragma: no-cache");
```

4.3. Cookies

Les cookies ont été introduits par la société Netscape dans le but de stocker des informations sur la machine cliente, ce qui permet de personnaliser un site en fonction de l'identité et des préférences du visiteur.

Ces cookies contiennent les informations que le client a bien voulu communiquer (en ayant, par exemple, répondu à un questionnaire sur ses goûts). Votre site peut alors très bien se servir de ces informations pour faciliter le parcours du visiteur. Un site de vêtements peut, par exemple, demander le sexe du client et le style de vêtements qu'il recherche pour ensuite le rediriger directement (et ce à chaque nouvelle connexion) vers le rayon sport homme si cela correspond au profil enregistré dans le cookie.

Plus d'informations sur les cookies
Les cookies sont définis dans la RFC 2109 disponible en anglais à l'adresse suivante : ***http://www.faqs.org/rfcs/rfc2109.html*** *ou encore* ***http://www.netscape.com/newsref/std/cookie_spec.html****.*

Généralités

Utilisation

Les cookies doivent être utilisés pour des informations de faible importance (pour compter, par exemple, le nombre de visites d'un client). Un bon site doit pouvoir se passer des cookies pour fonctionner : on peut les utiliser, mais cela ne doit pas mettre en péril le fonctionnement du site si l'utilisateur les refuse.

En effet, il faut avoir à l'esprit que les informations sont stockées côté client, d'où l'absence de garantie de pérennité des informations.

- Chaque navigateur a son propre système de gestion des cookies : deux navigateurs sur la même machine impliquent deux systèmes de gestion des cookies (information en double et problèmes de mise à jour...).
- Du fait que l'utilisateur peut lui-même changer la valeur des cookies, le programmeur n'a aucune garantie de "last value" (c'est-à-dire que l'on ne peut pas être certain de récupérer la valeur qui a été stockée précédemment).
- Enfin, la fonction de gestion des cookies peut être activée ou désactivée au niveau du navigateur ou simplement refusée par le visiteur.

Fonctionnement

Les instructions permettant la gestion des cookies sont placées dans les en-têtes HTTP. Lors de l'écriture d'un cookie, le serveur envoie la requête HTTP et le navigateur se charge d'écrire ou de modifier le cookie.

Stockage

Les cookies sont stockés de façon différente sur les divers navigateurs. Pour n'évoquer que des deux principaux navigateurs, Internet Explorer utilise un fichier recensant tous les cookies et un fichier par cookie, tandis que Netscape utilise un seul fichier pour tous les cookies.

Création

Pour que le serveur demande la création d'un cookie sur le poste client, l'en-tête de la réponse HTTP doit contenir une ligne avec la syntaxe suivante :

```
Set-Cookie : <NOM_COOKIE>=<valeur>; domain=<nom_de_domaine>; expires=<DATE>
```

Il existe d'autres attributs disponibles pour définir un cookie ; ils sont présentés dans le tableau suivant :

Tableau 4.5 : Attributs définissant un cookie

Attribut	Valeur	Type	Description
`NOM_COOKIE`	`VALEUR_COOKIE`	À moins de passer par l'encodage URL, le nom et la valeur d'un cookie ne peuvent pas contenir les caractères point-virgule '`;`', virgule '`,`', et espace ' '.	NOM_COOKIE est le seul attribut obligatoire.
`Expires`	`DATE`	Date au format : `Jour_en_anglais, JJ-Moi-AA HH:MM:SS GMT` Par exemple : `Saturday, 03-Aug-02 09:14:23 GMT` est une date valide.	`Expires` permet de définir une date de validité. Une fois cette date passée, la valeur du cookie ne doit plus être prise en compte et le cookie peut être effacé par le navigateur.
`Domain`	`Nom de domaine`	Adresse Internet contenant deux fois le caractère point (`.`). `Xxx.xxxxxxx.xxx` Par exemple : **www.toutestfacile.com** est une adresse valide ; **php.toutestfacile.com** également.	Si le nom de domaine est laissé vide (c'est généralement le cas), le nom du serveur appelant est assigné par défaut. Il n'est possible de spécifier que son propre nom de domaine ou de sous-domaine.
`Path`	`Chemin`	`/chemin/` Par exemple : `/php/` est valide.	Cet attribut permet de définir un sous-répertoire où le cookie est valide, afin de réduire son champ d'action. En effet, en spécifiant l'attribut `path`, le cookie ne sera accessible que dans le sous-répertoire défini.
`Secure`	Aucun		Cet attribut permet de spécifier que le cookie ne pourra être envoyé que si la connexion est sécurisée par SSL ou S-http.

Les limites des cookies

- *Un cookie ne pourra excéder 4 Ko.*
- *Un client ne pourra gérer plus de 300 cookies.*
- *Un serveur ne pourra créer plus de 20 cookies sur un client.*

Accès

Quand un client se connecte sur un site pour lequel il possède déjà des cookies, ceux-ci sont directement envoyés dans l'en-tête de la requête HTTP. Ce dernier contient alors une ligne ayant l'aspect suivant :

```
Cookie : <NOM1>=<valeur1> ; <NOM2>=<valeur2> ; ...
```

Le moyen de récupérer cette information dépendra alors du script utilisé (CGI, ASP, PHP, etc.).

Disponibilité
Un cookie n'étant disponible que lorsque le client le communique au serveur, un cookie défini dans un script ne sera disponible que des scripts chargés par la suite (et non pas dans la page courante).

En PHP

Pour envoyer un cookie, il suffit d'utiliser la fonction `setCookie()`.

setCookie()

Définit un cookie qui sera ajouté aux autres éléments de l'en-tête HTTP.

Syntaxe	`boolean setCookie(string $nomCookie [, string $valeurCookie [, int $dateExpiration [, string $chemin [, string $domaine [, boolean $securite]]]]])`
`$nomCookie`	Nom du cookie.
`$valeurCookie`	Valeur du cookie.
`$dateExpiration`	Date d'expiration du cookie (vous pouvez indiquer `NULL` si vous ne souhaitez pas spécifier ce paramètre). Il doit s'agir d'un 'timestamp' UNIX tel que peuvent retourner les fonctions `time()` ou `mktime()`.
`$chemin`	Répertoire de validité du cookie dans le site (vous pouvez indiquer `NULL` si vous ne souhaitez pas spécifier ce paramètre).
`$domaine`	Domaine de validité du cookie (vous pouvez indiquer `NULL` si vous ne souhaitez pas spécifier ce paramètre).
`$securite`	`TRUE` si vous souhaitez que ce cookie ne soit communiqué que lors de l'utilisation de connexions sécurisées (`SSL` ou `S-HTTP`), `FALSE` (valeur par défaut) sinon.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

Seul l'attribut `$nomCookie` est obligatoire. Si c'est le seul attribut défini, alors le cookie correspondant sera détruit sur la machine cliente.


REMARQUE

setCookie(), header(), même combat

Utiliser `setCookie()` revient à spécifier des données de l'en-tête. La remarque que l'on a vue pour la fonction `header()` s'applique donc là aussi : aucun élément du document ne doit avoir été émis avant l'appel à `setCookie()`.


ATTENTION

Erreur fréquente

Les cookies doivent être effacés avec les mêmes paramètres que lors de leur création.


REMARQUE

Ordre d'appel

En PHP 4 les appels à `setCookie()` se font dans l'ordre naturel, de haut en bas, alors qu'en PHP 3 les appels se faisaient dans l'ordre inverse. Ainsi, pour effacer un cookie avant de déclarer une nouvelle valeur, il faut mettre l'insertion avant l'effacement en PHP 3 et faire le contraire en PHP 4.

Voici quelques exemples de création de cookies :

```
// exemple simple sans date de validité
setcookie("test", "ceci est un test");
// exemple d'un cookie valide pour une heure
setcookie("test", "ceci est un test", time()+3600);
// exemple d'un cookie valide uniquement
// dans les pages sécurisées de http://www.toutestfacile.com/php
setcookie("test", "ceci est un test", NULL,
          "/php/", ".toutestfacile.com", TRUE);
```

Une fois le cookie défini, son contenu est disponible lors de l'appel des scripts suivants, directement dans `$_COOKIE["nomDuCookie"]` (ou dans `$HTTP_COOKIE_VARS ["nomDuCookie"]` pour les versions de PHP<4.1.0).


ATTENTION

Du passé faisons table rase

Dans les versions de PHP antérieures à 4.2.0, la configuration par défaut fixait le paramètre `register_globals` du fichier php.ini à `on` (activé). Pour des raisons de sécurité (évoquées dans le chapitre traitant des variables externes) cela n'est plus le cas.

Avec l'option `register_globals` activée, le contenu du cookie était directement disponible dans une variable portant le nom du cookie (`$nomDuCookie`).


RENVOI

Vous pouvez vous reporter au chapitre "Prise en main" pour plus de détails sur le fichier php.ini.


REMARQUE

Un tour de magie à connaître

Par défaut, le fichier de configuration php.ini active l'option `magic_quote_gpc`*, et ceci pour les paramètres GET, POST et les cookies. Ceci a pour effet d'automatiquement "échapper" les apostrophes. Autrement dit, cela ajoute un anti-slash devant les apostrophes des paramètres passés par les méthodes GET, POST et les cookies.*

Si c'est très pratique pour stocker le paramètre en base de données, cela peut devenir gênant pour un affichage dans le document. Si vous souhaitez supprimer ces anti-slashes, il vous suffit d'appliquer la fonction `stripSlashes()` *sur ces valeurs.*

Il est possible de stocker directement plusieurs valeurs d'un tableau. Un cookie sera créé par entrée dans le tableau. Au moment de récupérer ces valeurs, il suffira d'y accéder comme pour n'importe quel autre tableau.

Par exemple, pour stocker ce tableau contenant trois valeurs :

```
setcookie ("tableau[un]", "valeur1");
setcookie ("tableau[deux]", "valeur2");
setcookie ("tableau[trois]", "valeur3");
```

Pour afficher ces valeurs, il suffit d'écrire :

```
while (list ($cle, $valeur) = each ($_COOKIE["tableau"])) {
    echo "$cle: $valeur<br>\n";
}
```


Exemple utilisant des cookies

Mise en situation

L'exemple est celui d'un magasin d'instruments de musique voulant offrir la possibilité à ses clients de mettre des articles dans un panier virtuel (les utilisateurs refusant les cookies ne pourront pas utiliser ce panier).

Les fonctionnalités de ce panier virtuel seront simples. Il faudra être en mesure d'ajouter des articles un à un, de vider le panier et de calculer le montant total des articles contenus dans le panier. Pour simplifier, on considérera que ce vendeur ne propose que trois articles : une guitare, une basse et une batterie (ce qui est suffisant pour monter un groupe).

Les informations liées au panier seront stockées dans des cookies sous la forme d'un tableau.

Agencement des fichiers

Pour cet exemple, nous utiliserons quatre fichiers :

- Une page d'accueil '`accueil.html`'.
- Une page d'initialisation '`initialisation.php`' qui sera en charge d'initialiser les cookies permettant la gestion du panier.
- Une page pour ajouter un article '`ajout_article.php`', cette page sera la page principale pour l'interface, et c'est là que l'utilisateur ajoutera des articles, videra son panier ou calculera le montant de celui-ci.

- Une page pour calculer le montant total du panier '`calcul_total.php`'. Ce fichier calculera la somme des montants des articles présents virtuellement dans le panier. L'utilisateur aura alors le choix d'ajouter d'autres articles ou de vider son panier.

Voici un graphe représentant les interactions entre les différents scripts.

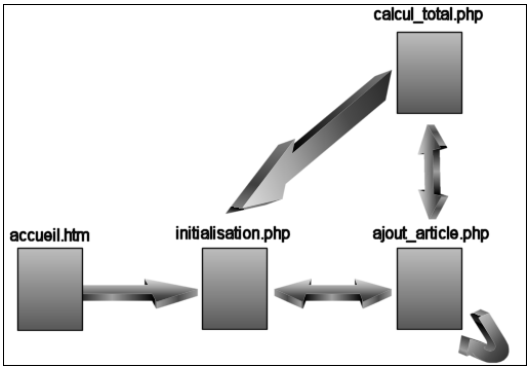


Figure 4.3 :
Les différents appels

- *accueil.htm* propose un lien vers *initialisation.php.*
- *initialisation.php* redirige vers *ajout_article.php* après avoir initialisé les données.
- *ajout_article.php* propose un lien vers *initialisation.php* pour réinitialiser les données.
- À chaque ajout d'article, *ajout_article* s'appelle lui-même pour se mettre à jour.
- *ajout_article* propose un lien vers *calcul_total.php.*
- Et *calcul_total.php* propose des liens vers *initialisation.php* (pour repartir de zéro) et *ajout_article.php* (pour compléter le panier).

Contenu des fichiers

accueil.html

Listing 4.1 : accueil.html

```
<html>
    <head>
      <title>Mon magasin de musique</title>
    </head>
    <body bgcolor="#FFFFFF">
      <p align="center"><b>Bienvenu chez MusicAGogo!!!</b></p>
      <p align="center">Votre panier est vide</p>
      <p align="center">
        <a href="initialisation.php">cliquer ici pour le remplir </a>
      </p>
    </body>
</html>
```

Le script d'accueil est une simple page au format HTML, avec un lien vers le script d'initialisation des cookies.

initialisation.php

Listing 4.2 : initialisation.php

```
<?php
    setCookie("panier[guitare]", 0);
    setCookie("panier[basse]", 0);
    setCookie("panier[batterie]", 0);
    header("Location: ajout_article.php");
?>
```

Le script d'initialisation met à zéro le tableau qui représentera notre panier. Une fois les trois valeurs du tableau initialisées, le navigateur du client est redirigé vers la page de l'interface principale.

Comme nous l'avons dit précédemment, la gestion des cookies est appelée avant toute autre fonction d'en-tête.

ajout_article.php

Listing 4.3 : ajout_article.php

```
<?php
    $panier=$_COOKIE["panier"];
    switch (@$_GET["ajout"]) {
        case "guitare":
            $panier["guitare"]++;
            setCookie("panier[guitare]", $panier["guitare"]);
            break;
        case "basse":
            $panier["basse"]++;
            setCookie("panier[basse]", $panier["basse"]);
            break;
        case "batterie":
            $panier["batterie"]++;
            setCookie("panier[batterie]", $panier["batterie"]);
            break;
    }
?>

<html>
  <head>
    <title>Mon magasin de musique</title>
  </head>
  <body bgcolor="#FFFFFF">
    <table border="4" cellspacing="4" cellpadding="4" align="center">
      <tr align="center">
```

```
        <td>Ajouter</td>
        <td>Votre commande</td>
      </tr>
      <tr align="center">
        <td>
          <a href="ajout_article.php?ajout=guitare">Une guitare</a> (199E)
        </td>
        <td><?php echo $panier["guitare"]?> guitare(s)</td>
      </tr>
      <tr align="center">
        <td>
          <a href="ajout_article.php?ajout=basse">Une basse</a> (199E)
        </td>
        <td><?php echo $panier["basse"]?> basse(s)</td>
      </tr>
      <tr align="center">
        <td>
          <a href="ajout_article.php?ajout=batterie">Une batterie</a> (2499E)
        </td>
        <td><?php echo $panier["batterie"]?> batterie(s)</td>
      </tr>
    </table>
    <p align="center">
      <a href="initialisation.php">vider mon panier</a>
    </p>
    <p align="center">
      <a href="calcul_total.php">calculer le total</a>
    </p>
  </body>
</html>
```

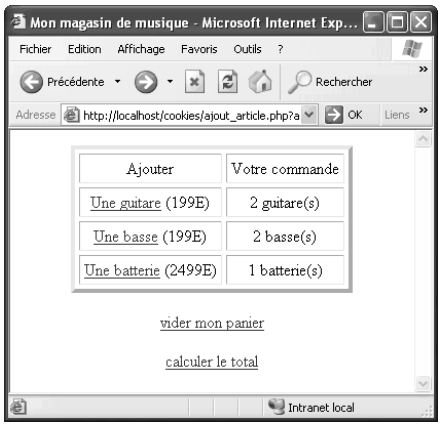


Figure 4.4 : *Interface principale*

Après avoir récupéré la valeur du tableau contenue dans les cookies, une comparaison est effectuée entre la valeur de la variable `$ajout` et les différentes chaînes de caractères possibles. Ensuite, la valeur est incrémentée puis stockée à nouveau dans le cookie correspondant.

Cette page fait des appels à elle-même, ce qui permet d'ajouter constamment des articles. Un lien vers *initialisation.php* est disponible, qui a pour effet de remettre à zéro le contenu du panier. Un autre lien vers le calcul du montant du panier est disponible.

Comme cela a déjà été signalé, la nouvelle valeur d'un cookie n'est disponible qu'après avoir rechargé la page ou changé de page. Pour afficher dans ce script le nombre exact d'articles contenus dans le panier, on ne peut faire appel à la valeur courante du cookie (par `$_COOKIE`) ; c'est donc le contenu de la variable du tableau `$panier` qui a été utilisé (après avoir été incrémenté puis stocké).

calcul_total.php

Listing 4.4 : calcul_total.php

```
<?php
    $panier = $_COOKIE["panier"];
    $total = 0;
    $total += $panier["guitare"]*199;
    $total += $panier["basse"]*199;
    $total += $panier["batterie"]*2499;
?>
<html>
    <head>
        <title>Mon magasin de musique</title>
    </head>
    <body bgcolor="#FFFFFF">
        <p align="center">Le total de votre panier:
                        <?php echo $total?> Euros.</p>
        <p align="center">
            <a href="ajout_article.php">Modifier mon panier</a>
        </p>
        <p align="center">
            <a href="initialisation.php">Vider mon panier</a>
        </p>
    </body>
</html>
```

Ce script très simple reprend les valeurs stockées dans le panier pour calculer le montant total de celui-ci.

Deux liens sont disponibles : l'un vers l'initialisation (remise à zéro) des variables, l'autre vers l'interface principale.

Figure 4.5 : *Montant total*

4.4. Sessions

Les sessions proposées dans PHP depuis la version 4 permettent, comme les cookies, de stocker des informations spécifiques à l'utilisateur.

Mais, alors que les cookies permettent de stocker des informations (en petite quantité) sur le poste du client pour une période pouvant aller de quelques secondes à plusieurs semaines, les sessions permettent de stocker autant d'informations que nécessaire. Pour cela, le serveur assigne au client un identifiant unique (appelé identifiant de session) lié à une instance de navigateur sur une machine (adresse IP) donnée. Du fait que le client peut à tout moment arrêter et relancer son navigateur ou changer d'IP (après s'être déconnecté du réseau Internet), la durée de vie de la session n'excède quasiment jamais une journée. Les sessions ne sont donc utilisées que pour conserver en mémoire des informations tout au long de la visite du client (mais pas plus longtemps). Par conséquent, il est même conseillé de mettre un terme à la session après une période donnée d'inactivité (de l'ordre de 20 mn, temps au-delà duquel on peut considérer que le visiteur a définitivement quitté le site) afin de libérer les ressources. Vous avez certainement rencontré ce genre de situation où l'on vous demande de vous identifier à nouveau.

En contrepartie, avec les sessions, vous pouvez être sûr que votre site fonctionnera quels que soient le navigateur et l'attitude du visiteur (rappelons qu'un visiteur peut refuser un cookie). L'autre différence principale réside dans le fait que les sessions sont stockées sur le serveur ; le contrôle des sessions est donc géré à 100 % par l'auteur des scripts (il ne peut y avoir de modification ou de suppression par le visiteur).

Comme cela a déjà été dit, à chaque utilisateur est attribué un identifiant de session. Afin de garder la trace de ce dernier, l'identifiant est transmis de page en page. Cela peut se faire de deux façons :

- Par le navigateur qui, à chaque appel au serveur, communiquera le cookie contenant l'identifiant de session que le serveur lui aura préalablement demandé de créer.
- Par le serveur qui, pour chaque page générée, complétera les URL relatives des liens (donc celles qui pointent vers le serveur lui-même, mais pas les liens vers un site tiers) avec un paramètre indiquant l'identifiant de session (c'est ce que l'on appelle l'"URL rewriting" ou la réécriture d'URL).

La première solution est la plus simple pour le serveur, mais elle ne peut être utilisée si le client refuse les cookies ; il faut alors se rabattre sur la seconde. Comme nous le verrons un peu plus loin, ceci peut se faire tant manuellement qu'automatiquement.

Pour chaque page nécessitant l'utilisation de variables de sessions, il faudra appeler la fonction `session_start()` afin de permettre au serveur de déterminer si un identifiant de session a déjà été assigné au visiteur. Si c'est le cas, il lui suffit de le récupérer soit dans le cookie soit depuis l'URL, tout comme les valeurs des différentes variables ; sinon, il doit le créer, et le serveur doit demander la génération d'un cookie. Si celle-ci échoue, il optera alors pour la réécriture d'URL.

session_start()

Réclame l'utilisation de variables de sessions au cours du script.

Syntaxe	`boolean session_start()`
`retour`	`TRUE`


REMARQUE

session_start() et cookie() : ami-ami
Il va de soi que si l'identifiant de session doit être stocké dans un cookie, alors les contraintes d'utilisation de `session_start()` sont les mêmes que celles de `cookie()`. Ainsi, `session_start()` ne peut plus être appelé une fois le début du document émis (i.e. une fois l'en-tête HTP envoyé).

Il sera possible de se passer de l'appel à `session_start()` si PHP est configuré de telle sorte que le paramètre `session.auto_start` de *php.ini* soit activé (i.e. mis à 1).


RENVOI

Vous pouvez vous reporter au chapitre "Prise en main" pour plus de détails sur le fichier php.ini.

À la manière des cookies, les variables de sessions sont disponibles directement dans le tableau `$_SESSION`. Pour accéder à une valeur donnée, il suffit donc d'appeler quelque chose comme `$_SESSION["maVariable"]` (ou `$HTTP_SESSION_VARS["maVariable"]` pour les versions de PHP inférieures à 4.1).


ATTENTION

Du passé faisons table rase
Dans les versions de PHP antérieures à 4.2.0, la configuration par défaut fixait le paramètre `register_globals` du fichier php.ini à `on` (activé). Pour des raisons de sécurité (évoquées dans le chapitre traitant des variables externes) cela n'est plus le cas.


ATTENTION

Avec l'option register_globals activée, le contenu de la variable de session était directement disponible dans une variable portant le nom de la variable de session (`$nomVariableSession`).


RENVOI

Vous pouvez vous reporter au chapitre "Prise en main" pour plus de détails sur le fichier php.ini.

Voici un exemple d'utilisation. Il s'agit tout simplement d'un compteur qui est incrémenté à chaque appel de la page (si la variable de session n'existe pas, c'est qu'il s'agit d'une nouvelle session et le compteur est initialisé à la valeur 0).

```
<?php
session_start();
if (!isset($_SESSION['maVariable'])) {
    $_SESSION['maVariable'] = 0;
}
else {
    $_SESSION['maVariable']++;
}
echo $_SESSION['maVariable'];
?>
```

Pour gérer les sessions de façon optimale, il est possible de configurer PHP à l'aide des nombreuses options offertes par le fichier de configuration.


RENVOI

Vous pouvez vous reporter au chapitre "Prise en main" pour plus de détails sur le fichier php.ini.

Ces options peuvent être visualisées par un simple appel à `phpinfo()`.

session

Session Support	enabled

Directive	Local Value	Master Value
session.auto_start	Off	Off
session.cache_expire	180	180
session.cache_limiter	nocache	nocache
session.cookie_domain	*no value*	*no value*
session.cookie_lifetime	0	0
session.cookie_path	/	/
session.cookie_secure	Off	Off
session.entropy_file	*no value*	*no value*
session.entropy_length	0	0
session.gc_maxlifetime	1440	1440
session.gc_probability	1	1
session.name	PHPSESSID	PHPSESSID
session.referer_check	*no value*	*no value*
session.save_handler	files	files
session.save_path	/tmp	/tmp
session.serialize_handler	php	php
session.use_cookies	On	On
session.use_trans_sid	1	1

Figure 4.6 : *phpinfo()*

Jusqu'à la version 4.1.1 (comprise), l'utilisation de la réécriture (automatique) d'URL ne pouvait se faire que si PHP avait été compilé avec l'option `--enable-trans-sid`. Il fallait en outre l'activer ou la désactiver avec l'option `session.use-trans-sid`, introduite dans la version 4.0.3 (à positionner à 1 pour l'activer, 0 sinon).

Depuis la version 4.2.0, l'utilisation de la réécriture (automatique) d'URL ne nécessite plus l'utilisation de directives de compilation. L'activation de ce procédé ne dépend plus que de `l'option session.use-trans-sid` (qui est, par défaut, activée).

La liste des balises HTML contrôlées est définie dans le paramètre `url_rewriter.tags` et, outre le fait que `iframe` n'apparaisse pas dans la liste, il faut garder à l'esprit que tous les liens créés par des fonctions Javascript (par exemple) ne profiteront pas de cette manipulation. Il faudra donc nécessairement ajouter manuellement l'identifiant de session à l'URL (si l'on souhaite que cela fonctionne avec les navigateurs refusant les cookies) pour ces liens.

Exemple utilisant des sessions

Mise en situation

Reprenons le même exemple que pour les cookies, c'est-à-dire celui d'un magasin de musique vendant trois instruments et proposant un panier virtuel aux visiteurs de son site Internet.

Contenu des fichiers

accueil.html

Listing 4.5 : accueil.html

```
<html>
    <head>
      <title>Mon magasin de musique</title>
    </head>
    <body bgcolor="#FFFFFF">
      <p align="center"><b>Bienvenu chez MusicAGogo!!!</b></p>
      <p align="center">Votre panier est vide</p>
      <p align="center">
        <a href="initialisation.php">cliquer ici pour le remplir </a>
      </p>
    </body>
</html>
```

Le script d'accueil est une simple page au format HTML, avec un lien vers le script d'initialisation des sessions.

initialisation.php

Listing 4.6 : initialisation.php

```
<?php
    session_start();
    $_SESSION['panier'] = array("guitare"=>0,
                                "basse"=>0,
                                "batterie"=>0);
    header("Location: ajout_article.php");
?>
```

Une fois la session existante rappelée ou créée, le tableau `panier` est initialisé et stocké dans la session ; le navigateur du client est ensuite redirigé vers la page d'ajout d'éléments au panier.

Ce script fonctionnera tel quel à condition que l'utilisateur accepte les cookies.

Si ce n'est pas le cas, il fonctionnera également si la réécriture (automatique) d'URL est activée (sachant que ce script ne contient que des URL relatives). Vous constaterez alors que les URL (qui s'affichent dans la barre du navigateur) ont été complétées pour ajouter un identifiant de session.

Dans les autres cas, si vous souhaitez que votre script fonctionne même si le visiteur refuse les cookies, il faut ajouter à l'URL la chaîne de caractères stockée dans la constante `SID` (pour réaliser manuellement ce que l'option `session.use_trans_sid = 1` permet). Ce qui donne :

```
header("Location: ajout_article.php?".SID);
```

La constante `SID` est composée de la concaténation du nom de l'identifiant de session (disponible via la fonction `session_name()` et qui par défaut vaut "`PHPSESSID`"), d'un signe égal et de la valeur de l'identifiant de session (disponible via la fonction `session_id()`). Ainsi, `SID` vaut `session_name()."=".session_id()`.

ajout_article.php

Listing 4.7 : ajout_article.php

```
<?php
    session_start();
    $panier=$_SESSION['panier'];
    switch (@$_GET["ajout"]) {
        case "guitare":
            $panier["guitare"]++;
            break;
        case "basse":
            $panier["basse"]++;
            break;
        case "batterie":
            $panier["batterie"]++;
            break;
    }
    $_SESSION['panier'] =
```

```
        array("guitare" => $panier["guitare"],
              "basse"   => $panier["basse"],
              "batterie"=> $panier["batterie"]);
?>

<html>
    <head>
         <title>Mon magasin de musique</title>
    </head>
    <body bgcolor="#FFFFFF">
        <table border="4" cellspacing="4"
               cellpadding="4" align="center">
            <tr align="center">
                <td>Ajouter</td>
                <td>Votre commande</td>
            </tr>
            <tr align="center">
                <td><a href="ajout_article.php?ajout=guitare">
                    Une guitare</a> (199E)
                </td>
                <td><?php echo $panier["guitare"]?> guitare(s)</td>
            </tr>
            <tr align="center">
                <td><a href="ajout_article.php?ajout=basse">
                    Une basse</a> (199E)
                </td>
                <td><?php echo $panier["basse"]?> basse(s)</td>
            </tr>
            <tr align="center">
                <td><a href="ajout_article.php?ajout=batterie">
                    Une batterie</a> (2499E)
                </td>
                <td><?php echo $panier["batterie"]?> batterie(s)</td>
            </tr>
        </table>
        <p align="center"><a href="initialisation.php">
                          vider mon panier</a></p>
        <p align="center"><a href="calcul_total.php">
                          calculer le total</a></p>
    </body>
</html>
```

Une fois la session rappelée, la variable `$panier` est récupérée, puis `$ajout` est testée afin d'incrémenter la bonne valeur. Une fois ce test effectué, la session est mise à jour avec les nouvelles valeurs.

Là encore, si votre serveur n'est pas configuré pour réaliser automatiquement la réécriture d'URL, vous devez compléter les URL avec la constante `SID` si vous voulez qu'il fonctionne également pour les clients refusant les cookies.

calcul_total.php

Listing 4.8 : calcul_total.php

```
<?php
    session_start();
    $panier = $_SESSION["panier"];
    $total  = 0;
    $total += $panier["guitare"]*199;
    $total += $panier["basse"]*199;
    $total += $panier["batterie"]*2499;
?>

<html>
    <head>
        <title>Mon magasin de musique</title>
    </head>
    <body bgcolor="#FFFFFF">
        <p align="center">
            Le total de votre panier: <?php echo $total?> Euros.
        </p>
        <p align="center">
            <a href="ajout_article.php">Modifier mon panier</a>
        </p>
        <p align="center">
            <a href="initialisation.php">Vider mon panier</a>
        </p>
    </body>
</html>
```

Ce script est plus simple à comprendre que le précédent. La session est récupérée puis la variable `$panier` en est extraite. Bien entendu, si besoin est, il faut prendre soin, là encore, de passer l'identifiant de session en paramètre.

Stockage personnalisé des variables de sessions

Par défaut les variables de sessions sont stockées dans des fichiers temporaires (comme l'indique le paramètre `session.save_handler` du fichier *php.ini* fixé à la valeur "`files`"). En modifiant ce paramètre à `user`, il est possible de redéfinir soi-même la façon dont seront gérées les sessions.

Vous pouvez vous reporter au chapitre "Prise en main" pour plus de détails sur le fichier php.ini.

Il est, par exemple, possible de stocker les paramètres dans une base de données (l'exemple que nous donnerons ici utilisera une base de données MySQL).

Vous pouvez vous reporter au chapitre "L'utilisation des bases de données" pour en savoir plus sur l'utilisation de MySQL avec PHP.

Pour cela, nous créerons une table avec trois champs : l'identifiant de session, la date d'expiration, et la valeur des variables de sessions. Notez bien que les variables de sessions seront automatiquement sérialisées et désérialisées (autrement dit, toutes les valeurs seront regroupées en une chaîne de caractères et inversement. Voir plus loin les fonctions `session_encode()` et `session_decode()`) par PHP).

Voici la syntaxe SQL pour la table `sessions`.

```
CREATE TABLE sessions (
    idsession CHAR(32) NOT NULL,
    expiration INT(11) NOT NULL,
    valeur TEXT NOT NULL,
    PRIMARY KEY (idsession)
);
```


Pour gérer les sessions, PHP appelle six fonctions. Pour personnaliser l'utilisation des sessions, il faut donc créer six fonctions auxquelles PHP fournira les paramètres, et indiquer à PHP d'utiliser ces fonctions en lieu et place des fonctions par défaut.

```
boolean ouvrirSession($cheminSession, $nomSession)
```

Cette fonction permet d'initialiser la gestion de sessions. En paramètre, l'on trouve le chemin (dans le cas d'une gestion par fichiers) où stocker les sessions. Il s'agit en fait du paramètre `session.save_path` du fichier *php.ini* qui peut contenir n'importe quelle information sans que ce soit nécessairement un chemin. `$nomSession` est le nom de la session tel qu'il est défini dans le fichier de configuration de PHP sous le nom `session.name` (par défaut, c'est `PHPSESSID`).

Elle devra retourner `TRUE` en cas de succès, `FALSE` sinon.

Dans notre exemple, cette fonction servira à établir la connexion avec la base de données.

```
boolean fermerSession()
```

Cette fonction permet de libérer d'éventuelles ressources occupées pour la gestion des sessions.

Elle devra retourner `TRUE` en cas de succès, `FALSE` sinon.

Dans notre exemple utilisant une connexion persistante, rien n'aura besoin d'être fait dans cette fonction. Sinon, nous aurions pu envisager de clore la connexion.

```
boolean ecrireSession($idSession, $valeur)
```

C'est à cette fonction que l'on va indiquer comment stocker les variables de sessions. `$idSession` est l'identifiant de session, qui servira à stocker la session (il pourra servir à définir le nom du fichier, qui sera donc unique, ou une entrée en base de données...). `$valeur` est la valeur à stocker dans la session ; elle est fournie sérialisée.

Elle devra retourner `TRUE` en cas de succès, `FALSE` sinon.

Dans notre exemple, c'est ici qu'il faut faire l'insertion en base de données en faisant attention de créer une nouvelle entrée s'il n'existe pas d'entrée pour cet identifiant, ou de modifier l'entrée existante.

```
mixed lireSession($idSession)
```

La fonction de lecture a pour paramètre un identifiant de session. Il faut retourner la valeur (sérialisée) sauvegardée, ou `FALSE` en cas d'erreur.

Dans l'exemple, il suffira de faire une recherche dans la table de la valeur correspondant à la l'identifiant de session fourni.

```
boolean detruireSession($idSession)
```

Cette fonction détruit la session dont l'identifiant est passé en paramètre.

Elle devra retourner `TRUE` en cas de succès, `FALSE` sinon.

Dans notre exemple, il suffira de supprimer l'entrée correspondant à l'identifiant passé en paramètre.

```
boolean nettoyerSession($dureeVie)
```

C'est la fonction qui sera appelée aléatoirement pour nettoyer les sessions invalides. La durée de vie, définie dans le fichier de configuration par le paramètre `session.gc_maxlifetime`, est passée en paramètre.

Elle devra retourner `TRUE` en cas de succès, `FALSE` sinon.

Dans notre exemple, il suffira de supprimer toutes les entrées dont la date d'expiration est passée.

Enfin, pour indiquer à PHP d'utiliser les fonctions ainsi définies, vous devrez faire appel à la fonction `session_set_save_handler()`.

session_set_save_handler()

Indique à PHP d'utiliser des fonctions personnalisées pour la gestion des sessions.

Syntaxe	`void session_set_save_handler(function $ouvrirSession, function $fermerSession, function $lireSession, function $ecrireSession, function $detruireSession, function $nettoyerSession)`
`$ouvrirSession`	Nom de la fonction à appeler à l'ouverture de la session (cf. ci-avant).
`$fermerSession`	Nom de la fonction à appeler à la fermeture de la session (cf. ci-avant).
`$lireSession`	Nom de la fonction à appeler à la lecture du contenu d'une session.
`$ecrireSession`	Nom de la fonction à appeler à la sauvegarde du contenu d'une session.
`$detruireSession`	Nom de la fonction à appeler à la destruction du contenu d'une session.
`$nettoyerSession`	Nom de la fonction à appeler pour détruire les sessions périmées.

session_mysql.php

Listing 4.9 : session_mysql.php

```
<?php
define("HOTE", "localhost");
define("UTILISATEUR", "root");
define("MOT_DE_PASSE", "MotDePasse");
define("BASE_DE_DONNEES", "biblephp");

define("DUREE_VIE_SESSION", get_cfg_var("session.gc_maxlifetime"));

$connexionSession = "";

function ouvrirSession($cheminSession, $nomSession)
{
    global $connexionSession;
    $connexionSession = mysql_pconnect(HOTE, UTILISATEUR, MOT_DE_PASSE)
    or die("Impossible de se connecter à la base de données");

    mysql_select_db(BASE_DE_DONNEES, $connexionSession)
    or die("Base de données introuvable");
    return TRUE;
}

function fermerSession()
{
    return TRUE;
}

function ecrireSession($idSession, $valeur)
{
    global $connexionSession;
    if (mysql_query("INSERT INTO sessions VALUES
                    ('$idSession',
                     ".(time()+DUREE_VIE_SESSION).",
                     '$valeur')", $connexionSession)) {
        return TRUE;
    } else {
        return mysql_query("UPDATE sessions SET expiration =
               ".(time()+DUREE_VIE_SESSION).", valeur = '$valeur'
               WHERE idSession = '$idSession'", $connexionSession);
    }
}

function lireSession($idSession)
{
    global $connexionSession;
    $requete = mysql_query("SELECT valeur FROM sessions
                            WHERE idsession='$idSession'
                            AND expiration > " . time(),
```

```
                            $connexionSession);
    if (list($valeur) = mysql_fetch_row($requete)) {
        return $valeur;
    } else {
        return FALSE;
    }
}

function detruireSession($idSession)
{
    global $connexionSession;
    return mysql_query("DELETE FROM sessions
                        WHERE idsession='$idSession'",
                        $connexionSession);
}

function nettoyerSession($dureeVie)
{
    global $connexionSession;
    return mysql_query("DELETE FROM sessions
                            WHERE expiration < " . time(),
                            $connexionSession);
}

session_set_save_handler("ouvrirSession",
                         "fermerSession",
                         "lireSession",
                         "ecrireSession",
                         "detruireSession",
                         "nettoyerSession");
?>
```


REMARQUE

Adapter le script à votre environnement

Les quatre constantes `HOTE`, `UTILISATEUR`, `MOT_DE_PASSE` *et* `BASE_DE_DONNEES` *doivent être renseignées selon votre configuration.*

Compte tenu des remarques précédentes, il n'est pas difficile d'écrire soi-même ce type de script.

La durée de vie d'une session est récupérée depuis le fichier de configuration grâce à la fonction `get_cfg_var()`. Cette valeur nous servira à déterminer la date d'expiration des sessions.

test.php

Ceci est un fichier permettant de tester la gestion des sessions telle qu'on vient de la définir.

Listing 4.10 : test.php

```
<?php
include("session_mysql.php");
```

```
session_start();

if (!isset($_GET["valeur"])) {
    $_GET["valeur"] = 0;
}
if ( (!isset($_GET["action"]))
   ||(!isset($_SESSION["variableSession"]))) {
    $_GET["action"] = "initialiser";
}

switch($_GET["action"]) {
    case "detruire":
        session_destroy();
        break;
    case "gc":
        nettoyerSession(get_cfg_var("session.gc_maxlifetime"));
        break;
    case "incrementer":
        $_SESSION["variableSession"]++;
        break;
    case "initialiser":
        $_SESSION["variableSession"] = 0;
        break;
}
?>
<html>
<head><title>Test</title></head>
<body>
    <p>
    Valeur courante:<?php echo $_SESSION["variableSession"]?>
    <br />
    <a href="test.php?action=incrementer">
        Incrémenter la variable de session</a>
    <a href="test.php?action=detruire">Destruction de la session</a>
    <a href="test.php?action=gc">Forcer le gc</a>
    </p>
</body>
```

Ce script très simple se contente d'afficher la valeur d'une variable de session, et de proposer trois liens : un pour l'incrémenter, un pour détruire la session et un pour forcer l'appel au garbage collector chargé de supprimer les sessions périmées.

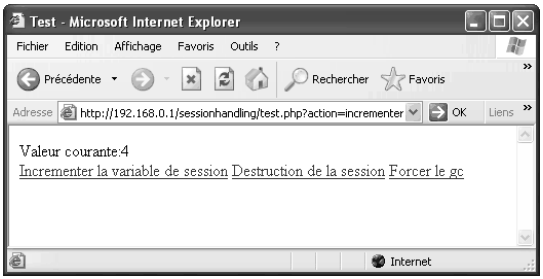


Figure 4.7 : *Interface de test*

Clore une session

Il n'est généralement pas possible de déterminer quand mettre fin à une session. En effet, pour cela, il faudrait savoir quand le visiteur a quitté le site. C'est pourquoi il faut généralement attendre la fin du délai d'expiration pour voir les sessions disparaître. Toutefois, si vous proposez sur votre site un lien *Déconnexion*, il vous sera possible de faire proprement appel à la fonction `session_destroy()` (Ce qui évitera en plus qu'un autre utilisateur partageant le même poste de travail puisse accéder au compte du précédent).

session_destroy()

Détruit les variables de la session en cours.

Syntaxe	`boolean session_destroy(void)`
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

Il est souhaitable de vider au préalable le contenu des variables de sessions. Ainsi le script de déconnexion pourra ressembler à :

```
<?php
    session_start();
    $_SESSION = array();
    session_destroy();
    echo "Au revoir, et à bientôt";
?>
```

Les autres fonctions

Comme cela a été précisé, les variables composant une session sont sérialisées avant d'être stockées (par défaut, dans un fichier), et désérialisées après avoir été récupérées. Pour réaliser manuellement cette opération, vous devez faire appel aux fonctions `session_code()` et `session_decode()`.

session_encode()

Sérialise la session (convertit l'ensemble des variables de sessions en une chaîne de caractères unique).

Syntaxe	`string session_encode(void)`
`retour`	Chaîne encodée avec le contenu de la session.

session_decode()

Désérialise une chaîne de caractères contenant des variables de sessions (recréant ainsi l'ensemble des variables de la session).

Syntaxe	`boolean session_decode(string $variable)`
`$variable`	Chaîne de caractères à désérialiser.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

D'autres fonctions, moins souvent utilisées, sont disponibles pour la gestion des sessions. Il est par exemple possible de remplacer l'utilisation des paramètres habituellement définis dans le fichier *php.ini* par des paramètres fixés par le script.

session_module_name()

Permet de définir ou de connaître le gestionnaire de sessions utilisé (habituellement défini par le paramètre `session.save_handler` de *php.ini*).

Syntaxe	`string session_module_name([string $gestionnaire])`
`$gestionnaire`	Indiquer (cf. `session_set_save_handler()`) : "`files`" si les données de session doivent être stockées "simplement" dans des fichiers temporaires. `user` si elles doivent être stockées en utilisant les fonctions personnalisées.
`retour`	Le gestionnaire de sessions utilisé à savoir `files`" ou "`user`".

session_save_path()

Permet de définir ou de connaître l'emplacement où seront sauvées les sessions (habituellement défini par le paramètre `session.save_path` de *php.ini*).

Syntaxe	`string session_save_path([string $chemin])`
`$chemin`	Chemin que l'on souhaite définir en tant que répertoire pour stocker les sessions.
`retour`	Nom du répertoire où sont stockés les fichiers temporaires de session.

session_name()

Permet de définir ou de connaître le nom du paramètre de l'URL contenant l'identifiant de session (habituellement défini par le paramètre `session.name` de *php.ini*).

Syntaxe	`string session_name([string $nomIdSession])`
`$nomIdSession`	Nouveau nom du paramètre d'identifiant de session.
`retour`	Nom de l'identifiant de session (par défaut `PHPSESSID`).

session_id()

Permet de définir ou de connaître l'identifiant de session attribué au visiteur.

Syntaxe	`string session_id([string $idSession])`
`$idSession`	Nouvel identifiant de session à utiliser.
`retour`	Identifiant de session utilisé.

session_regenerate_id()

Permet de régénérer un nouvel identifiant pour une même session. Cette fonction a été introduite avec PHP 4.3.2.

Syntaxe :	`boolean session_regenerate_id(void)`
`retour`	`TRUE` si un nouvel identifiant a été régénéré, `FALSE` sinon.

Il est également possible de connaître ou de fixer les paramètres du cookie chargé de sauvegarder l'identifiant de session sur la machine du visiteur (sans tenir compte des paramètres définis dans le fichier *php.ini*).

session_get_cookie_params()

Permet de récupérer les informations du cookie de session.

Syntaxe	`array session_get_cookie_params(void)`
`retour`	Tableau associatif contenant les informations du cookie. On y retrouve : "`lifetime`", la durée de vie du cookie. "`path`" le chemin où est stocké le cookie. "`domain`" le domaine sur lequel le cookie est valable. "`secure`" un booléen indiquant si le cookie ne doit être envoyé que par une connexion sécurisée.

session_set_cookie_params()

Permet de définir les paramètres du cookie de session.

Syntaxe	`void session_set_cookie_params(int $dureeVie [, string $chemin [, string $domaine]])`
`$dureeVie`	Nouvelle durée de vie en secondes (habituellement définie par le paramètre `session.cookie_lifetime` du fichier *php.ini*).
`$chemin`	Chemin où le cookie est valide (habituellement défini par le paramètre `session.cookie_path` du fichier *php.ini*).
`$domaine`	Domaine de validité du cookie (habituellement défini par le paramètre `session.cookie_domain` du fichier *php.ini*).

Les pages utilisant des variables de sessions présentent généralement un aspect différent d'un appel à l'autre, sans que leur URL ne soit modifiée. Pour que le visiteur puisse profiter des modifications apportées à la page, il faut absolument interdire l'utilisation du cache par le navigateur. C'est pour cela que, par défaut, les pages faisant appel aux sessions demandent également l'interdiction de la mise en cache. Mais ces paramètres peuvent être modifiés avec la fonction suivante :

session_cache_limiter()

Permet de définir ou de connaître la restriction du cache appliquée (habituellement définie par le paramètre `session.cache_limiter` de *php.ini*).

Syntaxe	`string session_cache_limiter([string $restriction_cache])`
`$restriction_cache`	Ce paramètre peut prendre les valeurs : "`nocache`" si l'on souhaite que le client ne mette pas la page dans le cache. "`public`" (ou "`private`" qui est légèrement plus restrictif) pour autoriser la mise en cache. "`private_no_expire`" (depuis la version PHP 4.2.0) permet de ne pas envoyer l'en-tête `Expire` qui causait des problèmes avec la restriction du cache.
`retour`	Retourne la restriction du cache en cours.

session_cache_expire()

Permet de définir ou de connaître la durée avant expiration du cache (habituellement définie par le paramètre `session.cache_expire` de *php.ini*). Cette fonction a été introduite à partir de la version 4.2.0.

Syntaxe	`int session_cache_expire([int $expirationCache])`
`$expirationCache`	Nouvelle valeur (en secondes) d'expiration du cache.
`retour`	Retourne la valeur actuelle d'expiration du cache.

Vous disposez aussi de :

session_write_close()

Termine la session et enregistre les variables de sessions, ce qui est automatiquement fait à la fin des scripts. Attention, dans ce cas, aucune réécriture d'URL ne sera effectuée.

Syntaxe	`void session_write_close(void)`

Les fonctions historiques

Les fonctions suivantes ne doivent plus être utilisées (pas avec le tableau `$_SESSION`, ni même d'ailleurs avec `$HTTP_SESSION_VARS`) :

`session_register()` permettait de définir une variable globale comme étant une variable de session. Ainsi, `register("maVariable");` faisait de `$maVariable` une variable de session. Avec l'utilisation de `$_SESSION["maVariable"]`, aucun doute n'est possible.

`session_unregister()` mettait simplement un terme à l'association entre la variable et une variable de session. Avec `$_SESSION`, il n'y a évidemment pas besoin d'équivalent.

`session_is_registered()` permettait de tester si une variable était une variable de session. Là encore, avec `$_SESSION`, il n'y a pas besoin d'équivalent.

`session_unset()` permettait de libérer le contenu des variables de sessions. Avec `$_SESSION`, il suffit de faire `$_SESSION = array()`.

4.5. Mise en cache avant émission des données

Les fonctions de base

Une série de fonctions de contrôle de sortie permet de gérer soi-même l'émission des données.

Habituellement, le fait de générer le document (avec les commandes `echo` ou `print` par exemple), et donc d'envoyer l'en-tête HTTP avant de faire appel à une fonction modifiant l'en-tête, aboutit inexorablement au message d'erreur suivant : "*Cannot add header information - headers already sent*". Il est possible d'éviter ceci en demandant de ne pas envoyer les éléments du document au moment où ils sont générés, mais de le faire une fois tous les éléments de l'en-tête définis.

Pour cela, nous disposons des fonctions `ob_start()` et `ob_end_flush()`.

ob_start()

Demande la mise en cache de ce qui va sur la sortie standard (ce qui est destiné au client).

Syntaxe `void ob_start([function $fonction])`

`$fonction` Fonction à appliquer sur le contenu du cache (cf. compression des données).

ob_end_flush()

Met fin à la mise en mémoire de la sortie et émet le contenu du cache vers le navigateur.

Syntaxe `void ob_end_flush(void)`

Ainsi, l'exemple suivant ne générera pas d'erreur malgré l'appel à `setCookie()` après un `echo`.

```
<?php
ob_start();
echo "J'envoie du texte sur la sortie avant de definir un cookie";
setCookie("cookie", "je defini un cookie apres avoir ecrit du texte");
ob_end_flush();
?>
```

Le fichier de configuration de PHP permet d'activer systématiquement le système de cache, grâce au paramètre `output_buffering`. Cela revient alors à placer `ob_start()` au début de chaque script, et `ob_end_flush()`, qui a pour effet d'afficher le contenu du cache à la fin.


RENVOI

Vous pouvez vous reporter au chapitre "Prise en main" pour plus de détails sur le fichier php.ini.


ATTENTION

Ne pas céder à la facilité

Même si l'utilisation du système de cache permet de mélanger allègrement génération du document et génération de l'en-tête HTTP, il est fortement conseillé (pour des raisons de performance) de se passer de l'utilisation de ces fonctions (dans ce cadre-là). D'autant qu'il est généralement facile de s'affranchir de la difficulté que cela pourrait engendrer. Il est de toute façon de bon usage de commencer un script par toutes les opérations de traitement avant les opérations de mise en page.

Compression des données

La plupart des navigateurs, dont Internet Explorer et Netscape, permettent la décompression transparente des données (les types de compression supportés sont précisés dans la variable externe `$_SERVER["HTTP_ACCEPT_ENCODING"]`). Si le serveur détecte que le navigateur du client supporte la compression Gzip (par exemple), il peut décider d'envoyer des données compressées à celui-ci, qui les décompressera avant de les afficher. Cette procédure sera totalement transparente pour l'utilisateur qui ne verra qu'un gain en terme de temps de téléchargement des pages.

Bien entendu, cette procédure demandera un petit peu plus de travail au serveur, mais si celui-ci est suffisamment puissant, son utilisation est avantageuse question rapidité.

Pour cela, il suffit de spécifier le paramètre "`fonction de compression`" de la fonction `ob_start()`. Cette fonction qui peut être une fonction personnalisée doit accepter un paramètre de type chaîne de caractères, et simplement en retourner la version compressée.

Voici un morceau de script à mettre en début de chacun de vos scripts pour utiliser la compression Gzip.

```
<?php
function compression($sortie)
{
    // Cette fonction ne fait que retourner la compression de la chaîne
    // de caractères précisée en paramètre.
    return gzencode($sortie);
}
// Vérification du support de gzip par le client
if (strstr($_SERVER['HTTP_ACCEPT_ENCODING'], 'gzip')) {
    // Début de mise en mémoire en fournissant comme paramètre
    // la fonction à appliquer sur la sortie
    ob_start("compression");
    // Prévient le navigateur que les données sont compressées.
    header("Content-Encoding: gzip");
}
?>
```

Gzip
Vous devrez avoir compilé PHP avec l'option --with-zlib pour pouvoir effectivement profiter de la compression Gzip.

La fonction `compression()` passée en paramètre à `ob_start()` est appelée juste avant l'envoi du résultat. Bien évidemment, ici, nous aurions pu nous passer de la fonction `compression()` et écrire directement `ob_start("gzencode")`. Qui plus est, depuis la version 4.0.4, PHP propose une fonction appelée `ob_gzhandler()` permettant de compresser les données en fonction du navigateur, en utilisant soit Deflate soit Gzip (voire ne compressant pas).

Tout le script précédent peut donc se réduire à :

```
<?php
    ob_start("ob_gzhandler");
?>
```

Si vous souhaitez utiliser systématiquement la fonction `ob_gzhandler()`, vous pouvez alors vous dispenser d'ajouter cette ligne au début de tous les scripts, simplement en modifiant le fichier de configuration *php.ini* pour avoir la ligne suivante :

```
output_handler = ob_gzhandler
```

Mais jusqu'où peut-on aller dans la simplification ? À ce stade, on ne peut plus faire grand chose...

L'exemple du premier script reste cependant utile dans certains cas, que vous pouvez imaginer vous-même. Il est en effet facultatif d'utiliser une fonction de compression, alors pourquoi ne pas, par exemple, appliquer une fonction qui rendra plus lisible le code HTML retourné pour déboguer vos scripts ? Il est également envisageable de ne produire que du XML et d'appliquer une feuille de styles au moment du rendu.

Optimisation des temps de réponse

Il est à tout moment possible de faire appel au contenu du cache, ce qui peut se révéler vraiment très intéressant.

L'utilisation la plus intéressante apparaît lorsque vous proposez un site contenant des pages dont le contenu peut prendre un certain temps à être généré (parce qu'il fait appel à une base de données, que le traitement des données est relativement long, etc.), mais qui varie peu dans le temps (dont le même résultat pourra être proposé à différents visiteurs sur une période donnée).

Dans ce cas, plutôt que de renouveler les opérations de traitement (et d'accès à la base de données) à chaque appel, il est préférable de générer le document (pour le premier visiteur) et de le stocker sous sa forme HTML (ou image, ou autre) afin de restituer le fichier tout prêt pour les visiteurs suivants. Il faudra simplement prendre soin de recréer ce document à intervalles réguliers coïncidant avec la fréquence de mise à jour des données.

Voyons maintenant comme procéder.

La fonction permettant de récupérer le contenu du cache s'appelle `ob_get_content()`.

ob_get_contents()

Récupère le contenu actuel du cache.

Syntaxe	`string ob_get_contents(void)`
`retour`	Contenu du cache, ou `FALSE` si la gestion du cache n'est pas activée.

Ainsi, afin de stocker le contenu d'un document dans un fichier, il suffit d'appeler le script suivant :

Listing 4.11 : cache_01.php

```
<?php
    ob_start();
```

```
    // Insérez ici, le code du script tel qu'il serait
    // sans l'utilisation de cache
    // Exemple echo "J'ai été mis en cache à ".strftime("%d/%m/%y %H:%M:%S");
    echo "J'ai été mis en cache à ".strftime("%d/%m/%y %H:%M:%S");

    $contenuCache = ob_get_contents();
    ob_end_flush();

    $fd = fopen("cache.html", "w");
    if (!$fd) die("Impossible d'ouvrir le fichier de cache");
    fwrite($fd, $contenuCache);
    fclose($fd);
?>
```

En plus d'être affiché, le contenu sera alors également stocké dans un fichier *cache.html*.

Pour que cela soit utile, reste à déterminer quand le fichier doit être recréé et quand il doit être affiché tel quel. Pour connaître la date de dernière mise à jour du fichier, nous pourrons faire appel à la fonction `filemtime()`. Il suffira alors de voir s'il date de plus de tant de temps (disons, à titre d'exemple, 2 mn).

Listing 4.12 : cache_01.php

```
<?php
    $fichierCache = "cache.html";

    if (@filemtime($fichierCache)<time()-2*60) {
        // Oulà... ça commence à dater
        ob_start();

        // Insérez ici, le code du script tel qu'il serait
        // sans l'utilisation de cache
        // Exemple echo "J'ai été mis en cache à ".
        //              strftime("%d/%m/%y %H:%M:%S");
        echo "J'ai été mis en cache à ".strftime("%d/%m/%y %H:%M:%S");

        $contenuCache = ob_get_contents();
        ob_end_flush();

        $fd = fopen($fichierCache, "w");
        if ($fd) {
            fwrite($fd, $contenuCache);
            fclose($fd);
        }
    } else {
        include($fichierCache);
    }
?>
```

Vous pouvez vous reporter aux sections "Dates" et "Fichiers" pour plus de détails sur les fonctions `strftime()`, `time()`, `filemtime()`, `fopen()`, `fwrite()` *et* `fclose()`.

Notez que, cette fois-ci, nous passons sous silence les cas où le fichier cache n'a pu être créé. En effet, il vaut mieux à ce moment-là ne pas polluer le document retourné (tant pis, nous ne profiterons tout simplement pas de l'effet de cache).

Notez également qu'en l'absence du fichier de cache (ou si celui-ci n'est pas lisible), la fonction `filemtime()` retournera un message d'erreur que nous masquons par un `@`. La valeur retour de `filemtime()` sera alors `FALSE` qui, converti en entier, vaut 0 et sera donc inférieur à la date limite ; le fichier cache sera donc bien recréé.

Il est également envisageable d'utiliser le cache juste pour récupérer ce qu'une fonction émet sur la sortie standard, sans que l'on veuille pour autant que cela apparaisse dans le document. Pour cela, vous devrez utiliser la fonction `ob_end_clean()`.

Gestion du cache interne

Par défaut, les éléments des documents à émettre vers un client sont mis dans un cache interne avant d'être envoyés au navigateur (ceci afin d'optimiser les transferts de données). Ce qui fait que les données ne seront pas mises à disposition du client dès qu'elles seront prêtes, mais dès que le cache sera plein (ou que l'exécution du script sera terminée).

En de rares circonstances, cela peut être frustrant. Imaginez que vous utilisez un script PHP pour effectuer un long travail de traitement. Vous aurez certainement envie de suivre son évolution en affichant régulière "tant de % réalisés". Malheureusement, il s'agit là d'une chaîne de caractères bien trop courte pour espérer voir le cache se remplir rapidement et être informé de l'avancée du traitement avant la fin du script.

Pour pallier cet inconvénient, il suffit de demander l'envoi du contenu de cache interne, même si celui-ci n'est pas plein. Ceci se réalise par un appel à la fonction `flush()` que vous appellerez chaque fois que vous voulez envoyer le contenu du cache.

Il est également possible de demander que l'appel à `flush()` se fasse systématiquement (dès qu'une nouvelle ligne est ajoutée au document). Pour cela, vous utiliserez la fonction `ob_implicit_flush()`.

ob_implicit_flush()

Active ou désactive l'envoi implicite. Quand l'envoi implicite est activé, les données sont envoyées dès que possible.

Syntaxe	`ob_implicit_flush([boolean $active])`
`$active`	`TRUE` (valeur par défaut) pour activer, `FALSE` pour désactiver.

Syntaxe	`boolean headers_sent(void)`
`retour`	`TRUE` si l'en-tête a déjà été émis, `FALSE` sinon.


REMARQUE

Partie de cache-cache
L'utilisation des fonctions `flush()` et `ob_implicit_flush()` n'implique pas la disponibilité immédiate des données au niveau de l'affichage du navigateur. En effet, après PHP, le serveur et le navigateur peuvent, eux aussi, décider de mettre les données dans un cache.

Quoi qu'il en soit, dans un environnement de travail de type "Linux + Apache + Mozilla" le script suivant,

Listing 4.13 : flush_02.php

```
<?php
    set_time_limit(0);

    ob_implicit_flush();

    for ($i=0; $i<=100; $i++)
    {
        echo "$i%<br />";
        sleep(1);
    }
?>
```

affiche 1 %, 2 %, etc. à intervalles réguliers d'une seconde, alors que la variante de ce script (*flush_01.php*) ne faisant pas appel à `ob_implicit_flush()` n'affiche rien pendant une longue période (probablement 100 secondes) avant d'afficher toutes les lignes d'un bloc.

Nous avons dû, ici, faire appel à `set_time_limit()` pour autoriser l'exécution d'un script de plus de 30 secondes.

Les autres fonctions

Ces fonctions sont également disponibles, mais n'ont pas été vues précédemment.

ob_get_length()

Retourne le nombre de caractères contenus dans le cache.

Syntaxe	`string ob_get_length(void)`
`return`	Longueur du contenu du cache, ou `FALSE` si la gestion n'est pas activée.

ob_flush()

Permet d'envoyer le contenu du cache au navigateur, puis de vider le cache sans pour autant mettre fin à la mise en cache. Cette fonction a été introduite dans la version 4.2.0 de PHP.

Syntaxe `void ob_flush(void)`

ob_clean()

Vide le contenu du cache sans pour autant mettre fin à la mise en cache. Cette fonction a été introduite dans la version 4.2.0 de PHP.

Syntaxe `void ob_clean(void)`

ob_get_level()

Retourne le degré d'imbrication des mises en cache (i.e. nombre d'appels à `ob_start()`). Cette fonction a été introduite dans la version 4.2.0 de PHP.

Syntaxe `int ob_get_level(void)`

`retour` Le degré d'imbrication des mises en cache.

Chapitre 5

Les techniques de programmation

5.1. Règles de codage

Contrairement à ce que l'on peut trouver avec le langage Java, les auteurs de PHP ne suggèrent pas véritablement de règles de codage. Cependant, en marge du développement du moteur PHP, les auteurs de PHP group contribuent également à la réalisation d'une bibliothèque de scripts PHP baptisée PEAR. Et, dans le cadre de ce projet, quelques règles de codage ont été édictées. Ce sont principalement ces recommandations que nous allons présenter ici, tout en les complétant et les améliorant (souvent en gardant l'esprit Java).

Ces règles de codage, souvent issues du simple bon sens, sont destinées à faciliter la lecture du code (notamment par un développeur autre que l'auteur principal du script), mais aussi à réduire les risques d'erreur de programmation.

Présentation du code

Il est ainsi suggéré :

- D'écrire des lignes de code ne dépassant pas 80 caractères (et idéalement ne dépassant pas 70 caractères) pour en faciliter la lecture à l'écran et sur papier.
- D'indenter le code avec quatre espaces (sans tabulations) afin de conserver la même mise en page quelle que soit la configuration de l'éditeur.
- De mettre une espace de part et d'autre d'un signe égal d'une affectation de valeur à une variable (voire plusieurs dans le cas d'une série d'affectations pour un alignement plus lisible).
  - Ce qu'il faut faire :

```
$var1      = fonction();
$variable2 = 21
```

  - Ce qu'il ne faut pas faire :

```
$variable2=21;
```

- D'insérer une espace entre les éléments des structures de contrôle et les parenthèses ouvrantes, afin de les différencier d'un appel de fonction.
  - Ce qu'il faut faire :

```
 if (condition) {
    code1;
}
```

  - Ce qu'il ne faut pas faire :

```
 if(condition) {
    code1;
}
```

  - Ce qu'il faut faire :

```
 if ((cond1) && (cond2) ) {
    code1;
}
```

- Ce qu'il ne faut pas faire :

```
if ((cond1)&&(cond2)) {
    code1;
}
```

- D'utiliser des accolades même pour les structures de contrôle ne contenant qu'une seule ligne. Ceci afin d'éviter tout oubli le jour où une ligne devra être ajoutée.
  - Ce qu'il faut faire :

```
if (condition1) {
    code1;
}
```

  - Ce qu'il ne faut pas faire :

```
 if (condition1)
    code1;
```

- De ne pas mettre d'espace entre le nom de la fonction et la parenthèse ouvrante précédant les paramètres.
  - Ce qu'il faut faire :

```
maFonction($param1)
```

  - Ce qu'il ne faut pas faire :

```
maFonction ($param1)
```

- De séparer chaque paramètre d'une fonction par une virgule suivie d'une espace (mais pas d'espace entre la parenthèse ouvrante et le premier paramètre).
  - Ce qu'il faut faire :

```
maFonction($param1, $param2)
```

  - Ce qu'il ne faut pas faire :

```
maFonction ($param1,$param2)
```

- De positionner l'accolade ouvrante de déclaration d'une fonction juste au dessous du "f" de `function` et de commencer le corps de la fonction après avoir indenté.
  - Ce qu'il faut faire :

```
function maFonction()
{
    return "blabla";
}
```

  - Ce qu'il ne faut pas faire :

```
function maFonction() {
    return "blabla";
}
```

Programmation

- Toujours utiliser les balises `<?php ?>` (et non `<? ?>` ou toute autre alternative).
- Les paramètres optionnels (avec des valeurs par défaut) des fonctions doivent être les derniers paramètres de la fonction (du moins couramment omis au plus couramment omis).
  - Ce qu'il faut faire :

```
function maFonc($p1, $p2 = "bla")
{
    return $p1.$p2;
}
```

  - Ce qu'il ne faut pas faire :

```
function maFonc($p1 = "bla", $p2)
{
    return $p1.$p2;
}
```

- Les fonctions doivent toujours retourner une valeur cohérente (i.e. en cas d'échec retourner clairement `FALSE` plutôt que de ne pas retourner explicitement de valeur. En cas d'appel à une fonction ne retournant pas spécifiquement de valeur, retourner `TRUE` en cas de succès, et `FALSE` sinon).
- Utiliser de préférence `include_once()` (plutôt que `include()`) pour les fichiers à inclure, mais dont on peut éventuellement se passer, et `require_once()` (plutôt que `require()`) pour les fichiers nécessaires au bon fonctionnement du script.
- Se tenir informé des problèmes de sécurité liés à l'utilisation de certaines fonctions (cf. include et upload de fichiers).

Noms de classes, fonctions, variables et constantes

Les règles de nommage de la bibliothèque `PEAR` ne sont pas tout à fait claires (du moins en ce qui concerne les classes et les variables). Mais une bonne pratique consiste à respecter à peu près les mêmes règles que pour Java (même si cela n'est pas le cas pour les fonctions natives de PHP et si, dans certains cas, on peut être amené à s'en écarter).

Les classes

Les noms de classes commencent par quelques lettres en majuscules indiquant le projet auquel elles appartiennent (ex. : SPP pour "Super Projet PHP") suivies d'un underscore et du nom décrivant le rôle de la classe. Ce nom ne comporte pas d'underscore et est essentiellement écrit en minuscules. Seules les premières lettres des mots composant le nom sont en majuscules. La lettre suivant l'underscore est, elle, une minuscule. Un nom de classe doit toujours commencer par une majuscule (ce qui est assuré ici par l'abréviation du nom du projet). Ce qui donne, par exemple :

```
class SPP_maClasse()
{
}
```

Les méthodes

Les noms de méthodes ne doivent pas commencer par une majuscule. Elles sont essentiellement écrites en minuscules. Seules les premières lettres des mots composant le nom sont en majuscules. Il est conseillé de distinguer les méthodes internes (uniquement utilisées par la classe) des méthodes externes (pouvant être appelées par le développeur) en faisant précéder les noms de méthodes internes par un underscore.

```
class SPP_superClasse
{
    function _fonctionInterne()
    {
        // commence par un underscore
    }

    function superFonction()
    {
        // pas de préfixe dans ce cas
    }
}
```


Les fonctions

Nous appliquerons aux fonctions (autres que les méthodes) les mêmes règles de nommage que celles appliquées aux classes.

```
function MA_superFonction()
{
}
```

Les variables

Les noms de variables suivent les mêmes règles que celles des noms de méthodes.

```
$maSuperVariable
$_maVariablePrivee
```

En revanche, pour les variables globales, les règles de nommage PEAR suggèrent de respecter les mêmes règles que pour les noms de classes, mais en les faisant précéder d'un underscore. (`$_PEAR_destructor_object_list` est cité en exemple, alors qu'il serait préférable d'utiliser le nom `$_PEAR_destructorObjectList`.)

Les constantes

Les noms des constantes sont en majuscules, et, dans ce cas, chaque mot est séparé par un underscore. Là encore, les noms doivent être précédés du nom, abrégé et en majuscules, de la bibliothèque.

```
define("SPP_SUPER_CONSTANTE", 20);
```

Commentaires

Ah ! les commentaires... Il faut distinguer deux types de commentaires. Il y a les commentaires destinés aux personnes qui vont devoir utiliser les classes et fonctions. Il s'agit dans ce cas de décrire l'objectif de la classe ou fonction ainsi que son interface (description des paramètres d'entrée et sortie). Puis il y a les commentaires destinés aux personnes qui seront chargées de la maintenance du code. Dans ce cas, il s'agit plutôt de décrire les algorithmes et les diverses subtilités qui ont été mis en place pour que la fonction fasse ce que l'on attend d'elle.

Dans le premier cas, les développeurs de `PEAR` suggèrent d'inclure dans les scripts des commentaires respectant la convention PHPDoc (fortement inspirée de JavaDoc).

Dans le second cas, il est suggéré d'utiliser soit la convention `/* */`, soit `//` (au choix du programmeur, mais pas `#`).

PHPDoc(umentor)

Installation

PHPDoc est plus ou moins un standard d'écriture repris du monde Java. Dans PEAR, on retrouve deux implémentations, une appelée PHPDoc et l'autre PHPDocumentor. PHPDocumentor est le générateur de documentation recommandé pour diverses raisons telles que la génération de fichier PDF, l'absence d'utilisation de base de données, possibilité de définir ses propres balises… En outre PHPDoc n'est plus en développement.

Pour installer PHPDocumentor une fois PEAR installé il suffit de taper `pear install phpdocumentor`.

PHP4/5
Pour utiliser PHPDocumentor avec PHP5 il vous faudra utiliser au minimum la version 1.3.0 autrement toute version est utilisable pour PHP4.

Syntaxe des commentaires

Tout comme pour JavaDoc, les lignes de commentaire doivent précéder les déclarations de classes, fonctions, etc. selon le schéma suivant :

```
/**
 * Description succincte
 *
 * Description détaillée
 *
 * @param    <type du parametre>    description
 */
```

Ces commentaires doivent apparaître avant les instructions `class` (déclaration d'une classe), `function` (déclaration d'une fonction), `var` (déclaration d'une variable), `define` (déclaration d'une constante), `include`, `include_once`, `require`, `require_once` (inclusion de fichiers).

Les mots-clés valides sont :

- `@access` suivi de "`private`" ou "`public`" pour indiquer si une classe, une fonction ou une variable est privée ou publique (par défaut, elles seront considérées comme privées).
- `@author` suivi du nom de l'auteur et éventuellement de l'e-mail mis entre < et > (ne peut pas être utilisé pour `include`, `include_once`, `require` et `require_once`). Le champ e-mail est censé être optionnel. En pratique, s'il n'est pas mis, le nom de l'auteur n'apparaît pas.
- `@const` suivi du nom de la constante et éventuellement de la description (valable uniquement pour `define`).
- `@deprecated` suivi d'un commentaire précisant depuis quand (date ou version) l'élément commenté est déprécié (ne peut être utilisé pour `include`, `include_once`, `require` et `require_once`).
- `@global`, pour préciser le rôle d'une variable globale utilisée dans une fonction, suivi du type de la variable (pour les objets, précisez "`object` " suivi du nom de l'objet), lui-même suivi du nom de la variable, enfin éventuellement suivi de la description (valable uniquement pour `function`).
- `@package`, pour préciser à quelle bibliothèque appartient une classe, suivi du nom de la bibliothèque (valable uniquement pour `class`).
- `@param`, pour préciser le rôle d'un argument d'une fonction, suivi du type de la variable (pour les objets précisez "`object` " suivi du nom de l'objet), suivi du nom de la variable, éventuellement suivi de la description (valable uniquement pour `function`).
- `@return`, pour préciser le contenu de la valeur retournée par une fonction, suivi du type de la variable (pour les objets précisez "object " suivi du nom de l'objet), éventuellement suivi de la description (valable uniquement pour `function`).
- `@see`, pour faire référence à une autre fonction, une autre méthode de la classe ou une variable, suivi du nom de l'élément pointé. Le nom de la fonction doit inclure les parenthèses ouvrante puis fermante. Cela insère alors un lien hypertexte dans la documentation (ne peut être utilisé pour `include`, `include_once`, `require` et `require_once`).
- `@since`, pour indiquer la date/version d'introduction de l'élément, suivi de la date ou de l'identifiant de version (ne peut être utilisé pour `include`, `include_once`, `require` et `require_once`).
- `@static`, pour indiquer qu'une méthode peut être considérée comme statique (appel `MonObjet::maMethode()` possible). Valable uniquement pour `function`.
- `@throws`, pour indiquer quelles erreurs peuvent être levées.
- `@var`, pour indiquer le rôle d'une variable, suivi du type de la variable (pour les objets précisez "`object`" suivi du nom de l'objet), suivi du nom de la variable, éventuellement suivi de la description (valable uniquement pour `var`).
- `@version`, pour indiquer le numéro de version, suivi d'une chaîne de caractères totalement libre indiquant le numéro de version.


REMARQUE

Un seul ou plusieurs fichiers
Contrairement à PHPDoc, avec PHPDocumentor il est possible de définir plusieurs classes dans un même fichier mais cela reste tout de même déconseillé pour des raisons de lisibilité (sauf cas particuliers).

Listing 5.1 : sources/phpdoc_demo01.php

```
<?php

    /**
     * @const SPP_maConstante Une bien belle constante
     */
    define("SPP_MA_CONSTANTE", 20);

    $_SPP_maVariableGlobale = "SPP";

    /**
     * Classe super pratique
     *
     * Là, je pourrais décrire ce que fait cette
     * classe mais comme je n'en ai aucune idée
     * je m'abstiendrais.
     *
     * @author Damien HEUTE <damien@toutestfacile.com>
     * @package SuperProjetPHP
     * @access public
     * @version 2.0
     * @since 2002-07-01
     */
    class SPP_maClasse
    {
        /**
         * @var string monParametre Paramètre de sauvegarde
         */
        var $monParametre;


        /**
         * @param monObjet object SPP_autreClasse Object a sauvegarder
         * @return boolean TRUE si OK, FALSE sinon
         * @see vieilleFonction()
         */
        function sauve($monObjet)
        {
            /**
             * @global string Une variable globale.
             */
            global $_SPP_maVariableGlobale;
```

```
            return $monObjet->sauve($_SPP_mavariableGlobale.
                                    $this->monParametre);
        }

        /**
         * @static
         * @param texte string Texte a afficher
         */
        function affiche($texte)
        {
            echo $texte;
        }

        /**
         * @deprecated 2.3
         */
        function vieilleFonction()
        {
            // ne plus utiliser depuis version 2.3
        }
    }

     /**
      * maFonction
      *
      * @param parametre1 string blabla
      */
    function maFonction($parametre1)
    {

    }
?>
```


Listing 5.2 : sources/phpdoc_demo02.php

```
<?php

    /**
     * Une classe vraiment bidon
     *
     * @package SuperProjetPHP
     */
    class SPP_autreClasse
    {

    }

?>
```

En-tête de fichier
Dans le cadre du projet PEAR, un en-tête de fichier est suggéré. Cela étant fortement dépendant du projet, nous n'aborderons pas ce point ici.

Génération de la documentation

La génération de la documentation est très simple, elle peut se faire "en ligne" ou "hors ligne" c'est-à-dire avec les fichiers sources sur le serveur ou dans tout autre répertoire.

Il y a deux possibilités pour générer la documentation, soit en ligne de commande soit par une interface web. L'inconvénient de l'interface web est qu'il faut l'installer sur le serveur et qu'elle ne permet de générer de la documentation que sur les fichiers présents sur le serveur. Nous allons donc voir uniquement le mode en ligne de commande.

Entrons tout de suite dans le vif du sujet avec un exemple ou dans le répertoire courant il y a le répertoire `sources` qui contient vos fichiers sources (dans notre exemple nous reprenons les fichiers décrit précédemment) et que l'on veuille générer une documentation HTML dans un nouveau répertoire appelé `documentation`:

```
phpdoc –d sources –t documentation
```

Voila, le tour est joué, voici le contenu du répertoire `documentation`:

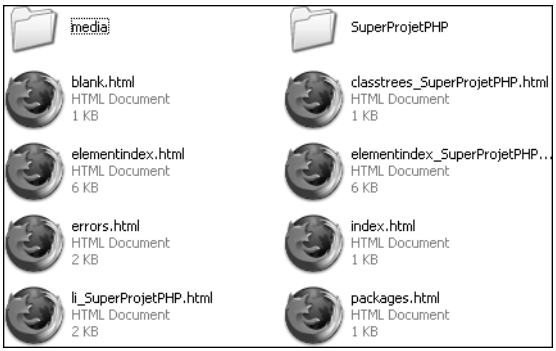


Figure 5.1 : *Contenu du répertoire documentation*

Et un aperçu de la documentation générée (voir fig. 5.2) :

Comme nous l'avons déjà préciser, PHPDocumentor permet de générer des fichiers PDF, ici nous n'avons rien préciser et donc le rendu obtenu est par défaut HTML avec la mise en page de PHPDocumentor.

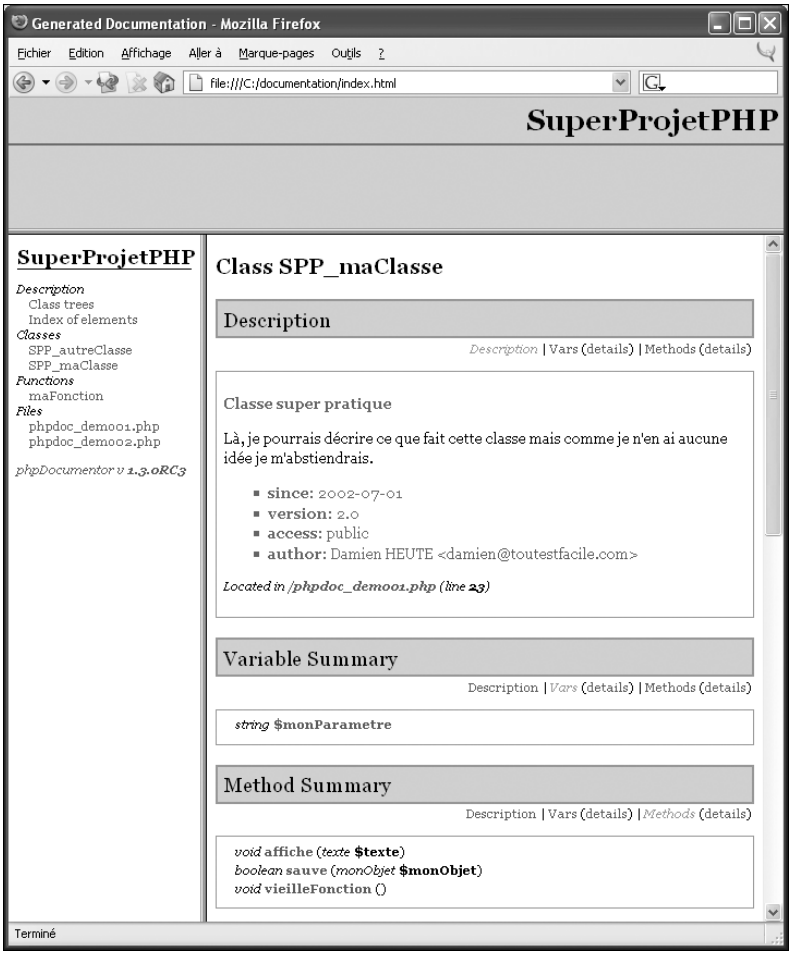


Figure 5.2 : *Aperçu de la documentation*

Voyons différentes options disponibles pour l'outil `phpdoc`:

- `-f` ou `--filename`: nom du ou des fichiers séparés par une virgule à analyser. (vous pouvez utiliser * ou ?)
- `-d` ou `--directory`: nom du ou des répertoires séparés par une virgule à analyser.
- `-t` ou `--target`: répertoire où créer la documentation. (il sera crée s'il n'existe pas déjà)
- `-i` ou `--ignore`: liste de fichiers à ignorer.
- `-q` ou `--quiet`: n'affiche pas d'informations lors de l'analyse.
- `-ti` ou `--title`: titre de la documentation ('Generated Documentation' par défaut)
- `-h` ou `--help`: permet d'obtenir une aide sur phpdoc.
- `-o` ou `--output`: défini le mode de génération. C'est cette option qui vous permet de générer de l'HTML, du PDF, du DocBook… Pour choisir il suffit d'aller dans le répertoire

`Converters` de PHPDocumentor et de définir le chemin vers le convertisseur que vous voulez. À ce jour il y a le choix entre: CHM:default, HTML:frames, HTML:Smarty, PDF:default, XML:DocBook. Il est également possible de créer votre propre mode.

Autres options
Il existe d'autres options mais moins souvent utilisées.

5.2. Séparation du code et de la mise en page

Pour des raisons de maintenance, ou tout simplement parce que les bons développeurs ne font pas nécessairement de bons designers (et réciproquement), il est généralement souhaitable de séparer le code de la mise en page. Ce principe général doit également s'appliquer au code PHP. Toutefois, s'il est évident que l'on ne va pas intégrer un gros bloc de traitement au milieu d'un script d'affichage, la mise en pratique de cette règle est souvent bien illusoire.

En effet, comme tout langage de programmation de pages web, PHP se situe à la frontière entre le code et la mise en page. Par conséquent, il n'est pas rare que les instructions d'affichage ne se bornent pas à afficher une image par-ci et le contenu d'une variable par-là. Plus fréquemment, les informations à afficher sont une liste d'informations (peut-être issues d'une base de données), chaque information pouvant avoir des attributs qui vont influencer la façon dont elle doit être affichée (par exemple, s'il s'agit d'une liste de noms de personnes, on pourra peut-être vouloir afficher une petite icône indiquant s'il s'agit d'un homme ou d'une femme). Dans ce cas, inévitablement, le script chargé de l'affichage devra contenir des instructions de boucle ainsi que des tests (pour avoir un affichage conditionnel selon les attributs). À moins, évidemment, que ce ne soit le script chargé du traitement de l'information qui retourne du code HTML (et gère donc les problèmes de mise en page).

À mon sens, la meilleure solution pour distinguer autant que possible le code de la mise en page consiste à faire judicieusement appel à des objets et fonctions stockés dans des scripts "bibliothèques". Nous ne pouvons toutefois pas taire les autres solutions (à base de modèles ou templates) proposées notamment au travers des bibliothèques `PHPLib` et `PEAR`.

Utilisation des objets et de l'instruction include

Le principe est simple : il suffit d'avoir un ou plusieurs scripts PHP ne contenant que des objets ou fonctions proposant des interfaces simples, et qui retourneront des données aisément manipulables par le ou les scripts chargés de la mise en page.

On pourra ainsi imaginer avoir des méthodes comme :

MaBD->rechercheVisiteurs()

Exemple de fonction qui rechercherait une liste de visiteurs dans une base de données à partir d'un critère quelconque, et retournerait `$nb` valeurs à partir de la `$debut`-ième.

Syntaxe	`array rechercheVisiteurs(string $critere, int $debut, $nb)`
`retour`	Tableau d'objet `Visiteur`, ou `FALSE` en cas d'échec.

Le script chargé de l'affichage pourrait alors avoir l'allure suivante :

```
<?php
    // Inclusion de toutes les bibliothèques nécessaires
    require_once("bibliotheque1_inc.php");

    // Initialisation de toutes les variables nécessaires
    $liste = MaBD::rechercheVisiteurs($critere, $debut, 10);
?>
<html>
<body>
<table>
<?php
    if ($liste) for ($i=0; $i<count($liste); $i++)
    {
?>
         <tr>
             <td>
                 <?php
                     if ($liste[$i]->sexe=="M") {
                       echo "<img src=\"homme.gif\">";
                     } else {
                       echo "<img src=\"femme.gif\">";
                     }
 ?>
             </td>
             <td><?php echo $liste[$i]->prenom; ?></td>
             <td><?php echo $liste[$i]->nom; ?></td>
         </tr>
<?php
    }
?>
</table>
</body>
</html>
```

Comme vous le voyez, nous sommes loin d'un script quasiment exempt de code. Mais il n'y a pas de miracle à espérer !

Il est également possible de faire appel aux possibilités offertes par les instructions `include()` et `require()` pour scinder une page complexe en plusieurs éléments (fichiers) dédiés à un sous-ensemble de la page. Ainsi, de nombreux scripts sont conçus selon le modèle :

```
<?php
    include("entete_inc.php");
    // Corps de la page
    include("pieddepage_inc.php");
?>
```

dans lequel le fichier *entete_inc.php* (plus couramment appelé *header.php*) contient généralement le titre du site, le logo, et, éventuellement, des barres de menu horizontale et latérale gauche (chacun de ces éléments pouvant aussi être dans des fichiers distincts). Le fichier *pieddepage_inc.php* (plus couramment appelé *footer.php*) contient, quant à lui, l'indication de copyright et, éventuellement, une barre de menu latérale droite.

L'utilisation de ces techniques offre de nombreux avantages :

- Elles sont très simples à mettre en œuvre.
- Elles facilitent la réutilisation du code.
- Elles sont relativement satisfaisantes en terme de séparation des rôles.

Utilisation des modèles (templates)

Avec la bibliothèque PHPLib

Installation

Vous devrez, dans un premier temps, vous procurer la bibliothèque `PHPLib` sur le site Internet **http://sourceforge.net/projects/phplib** (elle se trouve également sur le CD-ROM fourni). Il s'agit simplement d'un fichier *phplib-7.4-pre1.tar.gz* à décompresser dans le répertoire de votre choix.

Ce fichier contient juste une série de scripts PHP (inutile donc de recompiler PHP). Pour utiliser cette bibliothèque en toute liberté, il suffit d'ajouter le répertoire contenant la bibliothèque au chemin de recherche spécifié par le paramètre `include_path` du fichier *php.ini*. Cela peut également se faire en utilisant la fonction `set_ini()` (utilisée comme suit : `set_ini("include_path", $nouvelleValeurIncludePath);`) au début de chaque script faisant appel à la bibliothèque `PHPLib`.

Pour utiliser les modèles, il suffira alors d'inclure le script *template.inc* disponible dans le répertoire *php*. Si vous n'avez pas modifié le contenu d'`include_path`, peu importe, vous n'aurez qu'à préciser le chemin complet vers ce fichier lors de l'include.

Introduction

La séparation code/mise en page avec `PHPLib` prend un tout autre aspect.

Le designer pourra créer des fichiers semblables à des fichiers HTML classiques, dans lesquels il aura, en plus, intégré des mots-clés (entre accolades) ou défini des blocs (par des commentaires HTML). Ces fichiers HTML seront alors traités par un script écrit par le développeur faisant appel à la bibliothèque `PHPLib`, afin de remplacer les mots-clés et les blocs par leurs valeurs.

Exemple de fichier modèle (créé par le designer) :

Listing 5.3 : modeles/phplib_01.tpl

```
<html>
<head>
<title>{TITRE}</title>
</head>
```

```
<body>
<h3>{TITRE}</h3>
<table border="1">
    <!-- BEGIN blocLigne -->
    <tr>
        <td>{SEXE}</td>
        <td>{PRENOM}</td>
        <td>{NOM}</td>
    </tr>
    <!-- END blocLigne -->
</table>
<center>{COPYRIGHT}</center>
</body>
</html>
```

Exemple de fichier de substitution mot-clé/code (créé par le développeur) :

Listing 5.4 : phplib_01.php

```
<?php
    require_once("../phplib-7.4-pre1/php/template.inc");

    // Instanciation d'un objet Template
    // en précisant que :
    // * les fichiers de modele
    // sont stockés dans le répertoire modeles
    // * les mots clés non reconnus
    // seront conservés
    $modele = new Template("modeles", "keep");

    // Associe un identifiant au fichier modèle
    $modele->set_file("idModele", "phplib_01.tpl");

    // Définit les valeurs associées aux mots clés
    $modele->set_var(array("TITRE" => "Modèles avec PHPLib",
                           "COPYRIGHT" => "Copyright 2002"));

    // Extrait du fichier identifié par "idModele"
    // le block nommé "blocLigne" et le remplace par
    // le mot clé "lignes"
    $modele->set_block("idModele", "blocLigne", "lignes");

    // A titre d'exemple
    // remplace les mots clés du bloc ligne
    // par différentes valeurs

    $modele->set_var(array("SEXE" => "<img src=\"homme.gif\">",
                           "PRENOM" => "Pierre",
                           "NOM" => "Dupond"));
    $modele->parse("lignes", "blocLigne", true);

    $modele->set_var(array("SEXE" => "<img src=\"femme.gif\">",
```

```
                        "PRENOM" => "Anne",
                        "NOM" => "Durand"));
    $modele->parse("lignes", "blocLigne", true);

    $modele->set_var(array("SEXE" => "<img src=\"homme.gif\">",
                        "PRENOM" => "Jean",
                        "NOM" => "Bon"));
    $modele->parse("lignes", "blocLigne", true);


    // procède aux substitutions
    // et stocke le résultat dans une variable
    // "resultat"
    $modele->parse("resultat", "idModele");

    // affiche le résultat
    $modele->p("resultat");
?>
```

Ceci affichera alors :

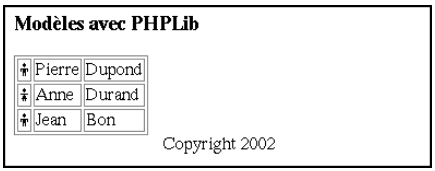


Figure 5.3 : *Exemple d'utilisation de modèles avec PHPLib*

Comme vous pouvez le constater, le designer n'a absolument pas besoin de connaître le langage PHP. En revanche, la tâche du développeur est sensiblement plus complexe (comme celle du serveur). Ce dernier doit apprendre un "nouveau langage", celui de la manipulation des modèles.

La base

L'utilisation des modèles PHPLib commence par l'instanciation d'un objet *Template* selon la syntaxe suivante :

Template

Objet modèle de PHPLib.

Syntaxe `Template Template(string $repertoireModeles, string $modeErreur)`

`$repertoireModeles` Répertoire où sont situés les fichiers de modèles.

`$modeErreur` Chaîne de caractères indiquant quel doit être le comportement lorsque des mots-clés inconnus sont rencontrés. Cet argument doit prendre une des valeurs suivantes :

"keep" si le mot-clé doit être restitué dans le flux de sortie.
"comment" si le mot-clé doit être mis en commentaire dans le flux de sortie.
"remove" (valeur par défaut) si le mot-clé doit être ignoré (et ne pas paraître dans le flux de sortie).

Toutes les manipulations à venir s'appuient sur des mots-clés (les chaînes de caractères intégrées au code HTML du modèle, y compris les noms de blocs) et des noms de variables (ni plus ni moins des chaînes de caractères représentant des portions du modèle). Il n'est pas rare que ces deux notions se confondent.

Même si, dans une utilisation "normale", cette méthode n'est appelée qu'à la fin du traitement, voici la méthode permettant d'afficher le contenu d'une variable. Dans notre cas, elle sera également utilisée pour démontrer les propriétés des différentes méthodes de l'objet *Template*.

Template->p()

Affiche le contenu de la variable indiquée. Cette fonction est généralement utilisée pour afficher le résultat final.

Syntaxe `void p(string $variable)`

`$variable` Nom de la variable dont on veut afficher le contenu.

L'utilisation la plus "primitive" des modèles consiste à remplacer des mots-clés (en absence de blocs) par des valeurs.

Cette opération s'effectue en deux étapes :

1. Définition des valeurs par lesquelles les mots-clés doivent être remplacés.
2. Remplacement proprement dit.

Template->set_var()

Donne une valeur à un ou plusieurs mots-clés.

Syntaxe `void set_var(string $motCle, string valeur)`

Syntaxe `void set_var(array $tableauAssociatif)`

`$motCle` Mot-clé auquel vous souhaitez donner une valeur.

`$valeur` Valeur associée au mot-clé.

`$tableauAssociatif` Tableau associatif contenant autant de couples (`$motCle => $valeur`) que de valeurs à définir.

Ainsi, la succession d'appels :

```
$modele->set_var("TITRE", "Mon TITRE");
$modele->set_var("COPYRIGHT", "Copyright 2002");
```

sera avantageusement remplacée par :

```
$modele->set_var(array("TITRE" => "Mon Titre",
                       "COPYRIGHT" => "Copyright 2002"));
```

La substitution proprement dite des mots-clés par les valeurs s'effectue via un appel à la méthode `parse()`.

Template->parse()

Remplace les mots-clés contenus dans une variable par leurs valeurs, et associe ou ajoute le résultat à un mot-clé.

Syntaxe	`void parse(string $motCle, $string $variable, boolean $modeAjout)`
`$motCle`	Mot-clé à remplacer ou compléter.
`$variable`	Variable pour laquelle les mots-clés doivent être remplacés par leurs valeurs.
`$modeAjout`	Mettre à `TRUE` si le mot-clé doit être complété, `FALSE` s'il doit être remplacé.

Dans le premier cas que nous étudions, la variable devra représenter le fichier modèle lui-même (c'est bien dans cet ensemble que les mots-clés doivent être remplacés par des valeurs). Il faut donc associer, au préalable, un nom de variable au fichier modèle. Pour cela, vous devez faire appel à la méthode `set_file()`.

Template->set_file()

Associe des noms de variables (raccourcis) à des noms de fichiers.

Syntaxe	`void set_file(string $variable, string $nomFichier)`
`$variable`	Variable associée au nom de fichier.
`$nomFichier`	Nom du fichier.

La méthode `set_file()` propose également l'interface suivante :

Template->set_file()

Associe des noms de variables (raccourcis) à des noms de fichiers.

Syntaxe `void set_file(array $tableauAssociatif)`

`$tableauAssociatif` Tableau associatif contenant autant de couples (`$variable => $nomFichier`) que d'identifiants à créer.

Ce qui nous permet de réaliser notre premier script de transformation de modèles ; script que nous appliquerons au modèle que nous avons vu précédemment.

Listing 5.5 : phplib_02.php

```
<?php
    require_once("../phplib-7.4-pre1/php/template.inc");

    // Instanciation d'un objet Template
    // en précisant que :
    // * les fichiers de modele
    // sont stockés dans le répertoire modeles
    // * les mots clés non reconnus
    // seront conservés
    $modele = new Template("modeles", "keep");


    // Définit les valeurs associées aux mots clés
    $modele->set_var(array("TITRE" => "Modèles avec PHPLib",
                           "COPYRIGHT" => "Copyright 2002"));

    // Associe un identifiant au fichier modèle
    $modele->set_file("idModele", "phplib_01.tpl");
    // Remplace les mots clés par leurs valeurs
    // et stocke le résultat dans une variable
    // "resultat"
    $modele->parse("resultat", "idModele");

    // affiche le résultat

    $modele->p("resultat");
?>
```

Dans ce cas, nous aurons alors comme résultat (au niveau du code HTML généré) :

```
<html>
<head>
<title>Modèles avec PHPLib</title>
</head>
<body>
<h3>Modèles avec PHPLib</h3>
<table border="1">
    <!-- BEGIN blocLigne -->
    <tr>
        <td>{SEXE}</td>
        <td>{PRENOM}</td>
        <td>{NOM}</td>
    </tr>
    <!-- END blocLigne -->
</table>
<center>Copyright 2002</center>
</body>
</html>
```

Figure 5.4 :
Code HTML issu de l'utilisation de modèles avec PHPLib

Vous pouvez constater que les mots-clés ont bien été remplacés par leurs valeurs, sauf évidemment ceux qui n'ont pas été traités par ce premier script, et qui appartiennent à des blocs dont nous allons, dès maintenant, détailler le fonctionnement.

Utilisation des blocs

Comme cela a été suggéré en introduction de ce chapitre (dans l'exemple), il est possible de définir des blocs de données.

Les blocs sont déclarés par des commentaires HTML contenant les instructions `BEGIN` et `END`.

```
<!-- BEGIN nomBloc -->
<!-- END nomBloc -->
```

La première opération liée à la manipulation de ces blocs consiste, généralement, à appeler l'instruction `set_block()`.

Template->set_block()

Remplace, dans une variable (représentant généralement un bloc), un bloc par un mot-clé et associe ce bloc à la variable de même nom.

Syntaxe	`void set_block(string $variable, string $nomBloc [, string $motCle])`
`$variable`	Nom de la variable dans laquelle est recherché le bloc.
`$nomBloc`	Nom du bloc (i.e. mot-clé précisé dans le commentaire définissant le bloc).
`$motClé`	Mot-clé venant en remplacement du bloc. Si ce paramètre est omis, il prendra la même valeur que `$nomBloc`.

C'est généralement dans le cadre de l'utilisation de blocs que le mode "ajout" de la méthode `parse()` prend tout son sens. Il est en effet alors possible de remplacer un (unique) mot-clé par une liste de valeurs. Voir le script *phplib_01.php* présenté en introduction de ce chapitre.

Diverses méthodes

Template->set_root()

Permet de modifier le chemin de recherche des modèles.

Syntaxe	`void set_root(string $repertoireModele)`
`$repertoireModele`	Nouveau chemin de recherche des modèles.

Template->set_unknowns()

Permet de modifier le comportement lorsque des variables inconnues sont rencontrées.

Syntaxe	`void set_unknowns(string $modeErreur)`
`$modeErreur`	Chaîne de caractères indiquant le comportement à tenir lorsque des mots-clés inconnus sont rencontrés. Cet argument doit prendre une des valeurs suivantes : "`keep`" si le mot-clé doit être restitué dans le flux de sortie. "`comment`" si le mot-clé doit être mis en commentaire dans le flux de sortie. "`remove`" (valeur par défaut) si le mot-clé doit être ignoré (et ne pas paraître dans le flux de sortie).

Template->subst()

Retourne la variable indiquée pour laquelle les mots-clés connus ont été remplacés par leurs valeurs. Les mots-clés inconnus seront laissés inchangés (quel que soit le mode d'erreur sélectionné).

Syntaxe	`string subst(string $variable)`
`$variable`	Nom de la variable à retourner.

Template->psubst()

Affiche le résultat que retourne `subst()`.

Syntaxe	`string psubst(string $variable)`
`$variable`	Nom de la variable à afficher.

Template->finish()

Retourne la variable indiquée pour laquelle les mots-clés connus ont été remplacés par leurs valeurs. Les mots-clés inconnus sont traités selon le mode d'erreur sélectionné.

Syntaxe	`string finish(string $variable)`
`$variable`	Nom de la variable à retourner.

Template->get_vars()

Retourne un tableau associatif des variables définies, les clés étant les noms des variables.

Syntaxe	`array get_vars()`
`retour`	Le tableau association "nom de variable" => "valeur".

Template->get_undefined()

Retourne un tableau (associatif) des mots-clés rencontrés n'ayant pas de valeurs associées.

Syntaxe	`array get_undefined()`
`retour`	Le tableau associatif "nom de variable" => "nom de variable".

haltMsg()

Il s'agit cette fois d'une fonction et non d'une méthode appelée accompagnée d'un message en cas d'erreur. Cette fonction peut être réécrite par vos soins.

Syntaxe	`void haltMsg(string $messageErreur)`
`$messageErreur`	Message d'erreur.

Avec la bibliothèque PEAR

Installation

L'installation de la bibliothèque `PEAR` ne nécessite quasiment aucun effort. En effet, celle-ci étant livrée avec PHP, il suffit d'ajouter le répertoire contenant la bibliothèque (ex. : */usr/local/lib/php* par défaut sous Linux) au chemin de recherche spécifié par le paramètre `include_path` du fichier *php.ini*. Cela peut également se faire en utilisant la fonction `set_ini()` (utilisée comme suit : `set_ini("include_path", $nouvelleValeurIncludePath);`) au début de chaque script faisant appel à la bibliothèque `PEAR`.

Nous supposerons par la suite que vous avez intégré `PEAR` au chemin de recherche spécifié dans *php.ini*. Pour utiliser les modèles, il suffira alors d'inclure le script *IT.php* disponible dans le répertoire *HTML*.

Utilisation

La bibliothèque `PEAR` (livrée avec PHP) propose quant à elle une solution fort similaire à celle proposée par `PHPLib`.

L'une d'elle, basée sur la classe `IntegratedTemplate` nécessite toutefois moins de manipulations de variables, ce qui simplifie, entre autres, l'utilisation des blocs.

L'exemple présenté en introduction de la bibliothèque `PHPLib` devient alors :

exemple de fichier modèle (créé par le designer) :

Listing 5.6 : modeles/pear_01.tpl

```
<html>
<head>
<title>{TITRE}</title>
</head>
<body>
<h3>{TITRE}</h3>
<table border="1">
    <!-- BEGIN blocLigne -->
    <tr>
        <td>{SEXE}</td>
        <td>{PRENOM}</td>
        <td>{NOM}</td>
    </tr>
    <!-- END blocLigne -->
</table>
</body>
</html>
```

Exemple de fichier de substitution mot-clé/code (créé par le développeur) :

Listing 5.7 : pear_01.php

```
<?php
    require_once("HTML/IT.php");

    // Instanciation d'un objet IntegratedTemplate
    // en précisant que :
    // * les fichiers de modele
    // sont stockés dans le répertoire "modeles"
    $modele = new IntegratedTemplate("modeles");

    // Charge le fichier modele
    $modele->loadTemplateFile("pear_01.tpl");

    // Defini les valeurs associées au mot clé
    $modele->setVariable("TITRE","Modèles avec PEAR");

    // Selectionne le bloc "blocLigne"
    // pour les prochaines manipulations
    $modele->setCurrentBlock("blocLigne");

    // A titre d'exemple
    // remplace les mots clés du bloc ligne
    // par différentes valeurs

    $modele->setVariable(array("SEXE" => "<img src=\"homme.gif\">",
                           "PRENOM" => "Pierre",
                           "NOM" => "Dupond"));
    $modele->parseCurrentBlock("blocLigne");

    $modele->setVariable(array("SEXE" => "<img src=\"femme.gif\">",
                           "PRENOM" => "Anne",
                           "NOM" => "Durand"));

    $modele->parseCurrentBlock("blocLigne");

    $modele->setVariable(array("SEXE" => "<img src=\"homme.gif\">",
                           "PRENOM" => "Jean",
                           "NOM" => "Bon"));
    $modele->parseCurrentBlock("blocLigne");

    // affiche le résultat
    $modele->show();
?>
```

La base

L'utilisation des modèles PEAR/IntegratedTemplate commence par l'instanciation d'un objet *IntegratedTemplate* selon la syntaxe suivante :

IntegratedTemplate

Objet modèle de PEAR/IntegratedTemplate.

Syntaxe `IntegratedTemplate IntegratedTemplate([string $repertoireModeles])`

`$repertoireModeles` Répertoire où sont situés les fichiers de modèles.

Toutes les manipulations à venir s'appuient sur des mots-clés ou noms de blocs intégrés au code HTML du modèle, que l'on pourra appeler "variable".

Même si, dans une utilisation normale, cette méthode n'est appelée qu'à la fin du traitement, voici la méthode permettant d'afficher le contenu d'une variable. Dans notre cas, elle sera également utilisée pour démontrer les propriétés des différentes méthodes de l'objet *IntegratedTemplate*.

IntegratedTemplate->show()

Affiche le contenu d'une variable (bloc ou mot-clé) indiquée. Cette fonction est généralement utilisée sans paramètre pour afficher le résultat final.

Syntaxe `void show([string $variable])`

`$variable` Nom de la variable (bloc ou mot-clé) dont on veut afficher le contenu.

Le chargement d'un modèle se fait via la méthode `setTemplate()` ou plus probablement `loadTemplateFile()`.

IntegratedTemplate->setTemplate

Charge un modèle basé sur une simple chaîne de caractères.

Syntaxe `boolean setTemplate(string $modele [, boolean $supprimeVariablesInconnues [, boolean $supprimeBlocsVides]]))`

`$modele` Le modèle.

`$supprimeVariablesInconnues` `TRUE` (par défaut) si vous souhaitez que les variables (mot-clés) non définis soient supprimés (en sortie), `FALSE` sinon.

`$supprimeBlocsVides` `TRUE` (par défaut) si vous souhaitez que les blocs vides soient supprimés (en sortie), `FALSE` sinon.

`retour` `TRUE` en cas de succès, `FALSE` sinon.

IntegratedTemplate->loadTemplateFile()

Charge un modèle stocké dans un fichier.

Syntaxe	`boolean loadTemplateFile(string $nomFichier [, boolean $supprimeVariablesInconnues [, boolean $supprimeBlocsVides]])`
`$nomFichier`	Nom du fichier modèle.
`$supprime VariablesInconnues`	`TRUE` (par défaut) si vous souhaitez que les variables (mots-clés) non définies soient supprimées (en sortie), `FALSE` sinon.
`$supprimeBlocsVides`	`TRUE` (par défaut) si vous souhaitez que les blocs vides soient supprimés (en sortie), `FALSE` sinon.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

L'utilisation la plus "primitive" des modèles consiste à remplacer des mots-clés (éventuellement contenus dans un bloc) par des valeurs.

Cette opération s'effectue en trois étapes :

1. Sélection du bloc ;
2. Définition des valeurs par lesquelles les mots-clés doivent être remplacés ;
3. Remplacement proprement dit.

La sélection du bloc se fait naturellement par un appel à la méthode `setCurrentBloc()`.

IntegratedTemplate->setCurrentBloc()

Sélectionne un bloc pour les manipulations à venir.

Syntaxe	`void setCurrentBloc([string $bloc])`
`$bloc`	Nom du bloc. Si ce paramètre est omis, c'est alors le modèle complet qui devient le bloc courant.

IntegratedTemplate->setVariable()

Donne une valeur à un ou plusieurs mots-clés.

Syntaxe	`void setVariable(string $motCle, string valeur)`
Syntaxe	`void setVariable(array $tableauAssociatif)`
`$motCle`	Mot-clé auquel vous souhaitez donner une valeur.
`$valeur`	Valeur associée au mot-clé.

`$tableauAssociatif` Tableau associatif contenant autant de couples (`$motCle => $valeur`) que de valeurs à définir.

Ainsi, la succession d'appels :

```
$modele->setVariable("TITRE", "Mon TITRE");
$modele->setVariable("COPYRIGHT", "Copyright 2002");
```

sera avantageusement remplacée par :

```
$modele->setVariable(array("TITRE" => "Mon Titre",
                           "COPYRIGHT" => "Copyright 2002"));
```

La substitution proprement dite des mots-clés par leurs valeurs s'effectue via un appel à la méthode `parseCurrentBlock()`.

IntegratedTemplate->parseCurrentBlock()

Remplace les mots-clés contenus dans le bloc courant par leurs valeurs.

Syntaxe	`boolean parseCurrentBlock(void)`
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

Diverses méthodes

IntegratedTemplate->setRoot()

Permet de modifier le chemin de recherche des modèles.

Syntaxe	`void setRoot(string $repertoireModele)`
`$repertoireModele`	Nouveau chemin de recherche des modèles.

IntegratedTemplate->parse()

Remplace les mots-clés contenus dans une variable par leurs valeurs.

Syntaxe	`boolean parse([string $variable] [, boolean $modeRecursif])`
`$variable`	Variable contenant les mots-clés à remplacer. Par défaut, le modèle complet.
`$modeRecursif`	À ignorer, au moins pour l'instant.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

IntegratedTemplate->get()

Retourne le contenu d'une variable pour laquelle les mots-clés ont été remplacés par leurs valeurs.

Syntaxe	`string get([string $variable])`
`$variable`	Variable dont on veut retourner le contenu. Si ce paramètre est omis, c'est tout le modèle qui est retourné.
`retour`	Contenu de la variable (ou bloc).

IntegratedTemplate->touchBlock()

Empêche la suppression d'un bloc vide, même si l'option a été positionnée.

Syntaxe	`boolean touchBlock(string $bloc)`
`$bloc`	Bloc à conserver.

Chapitre 6

Les fonctions mathématiques

Cette partie traite du calcul mathématique. Même si vous serez souvent amené à utiliser ces fonctions pour des opérations élémentaires, sachez qu'il est également possible d'effectuer des calculs de précision.

6.1. Les fonctions mathématiques et les constantes

La bibliothèque mathématique fournit des constantes et des fonctions qui vous serviront si vous souhaitez faire du calcul. Attention cependant : pour la gestion des grands nombres ou des nombres devant comporter de nombreuses décimales, vérifiez que les valeurs que vous manipulez peuvent être gérées soit par les entiers (`integer` entre -2147483648 et 2147483647) soit par les réels (`double` ; et respectivement `long` et `double` en C). Dans le cas contraire, vous devrez faire appel aux calculs de précision qui feront l'objet du chapitre suivant.

Constantes

La bibliothèque mathématique met à disposition dix-sept constantes utiles pour les calculs mathématiques.

Tableau 6.1 : Les constantes

Constante	Valeur	Description
`M_SQRT2`	1.41421356237309504880	Racine carrée de 2 (PHP4.0.2 et +)
`M_SQRT3`	1.73205080756887729352	Racine carrée de 3 (PHP4.0.2 et +)
`M_SQRT1_2`	0.70710678118654752440	1 sur racine carrée de 2 ou encore racine carrée de 2 sur 2 (PHP4.0.2 et +)
Trigonométrie		
`M_PI`	3.14159265358979323846	Pi
`M_PI_2`	1.57079632679489661923	Pi/2
`M_PI_4`	0.78539816339744830962	Pi/4
`M_1_PI`	0.31830988618379067154	1/Pi
`M_2_PI`	0.63661977236758134308	2/Pi
`M_SQRTPI`	1.77245385090551602729	Racine carrée de Pi (PHP4.0.2 et +)
`M_2_SQRTPI`	1.12837916709551257390	2 sur racine carrée de Pi (PHP4.0.2 et +)
Logarithme		
`M_E`	2.7182818284590452354	E
`M_LOG2E`	1.4426950408889634074	Logarithme binaire de e
`M_LOG10E`	0.43429448190325182765	Logarithme décimal de e
`M_LN2`	0.69314718055994530942	Logarithme népérien de 2
`M_LN10`	2.30258509299404568402	Logarithme népérien de 10
`M_LNPI`	1.14472988584940017414	Logarithme népérien de Pi (PHP4.0.2 et +)

Constante	Valeur	Description
Autre		
M_EULER	0.57721566490153286061	Constante d'Euler (PHP4.0.2 et +)

Fonctions

Test de validité

PHP propose quelques fonctions permettant de déterminer si une valeur (retournée par une fonction) est finie, infinie (cas d'une division par 0) ou indéfinie (cas d'une racine carré de -1). Vous disposez ainsi de :

is_finite()

Indique si une valeur est finie ou non.

Syntaxe	`boolean is_finite(double $valeur)`
`$valeur`	Valeur à tester.
`retour`	`TRUE` si la valeur est finie, `FALSE` sinon.

is_infinite()

Teste si une valeur est infinie.

Syntaxe	`boolean is_infinite($valeur)`
`$valeur`	Valeur à tester.
`retour`	`TRUE` si la valeur est infinie, `FALSE` sinon.

is_nan()

Teste si une valeur est indéfinie.

Syntaxe	`boolean is_nan($valeur)`
`$valeur`	Valeur à tester.
`retour`	`TRUE` si la valeur est indéfinie, `FALSE` sinon.

Listing 6.1 : is_xxx.php

```
<?php
    if (is_finite(3)) echo "3 est fini<br />";
        else echo "3 n'est pas fini<br />";
    if (is_finite(pow(0,-1))) echo "1/0 est fini<br />";
        else echo "1/0 n'est pas fini<br />";
    if (is_finite(sqrt(-1))) echo "racine carrée de -1 est fini<br />";
        else echo "racine carrée de -1 n'est pas fini<br />";
    if (is_infinite(3)) echo "3 est infini<br />";
        else echo "3 n'est pas infini<br />";
    if (is_infinite(pow(0,-1))) echo "1/0 est infini<br />";
        else echo "1/0 n'est pas infini<br />";
    if (is_infinite(sqrt(-1)))
            echo "racine carrée de -1 est infini<br />";
        else echo "racine carrée de -1 n'est pas infini<br />";
    if (is_nan(3)) echo "3 est indéfini<br />";
        else echo "3 n'est pas indéfini<br />";
    if (is_nan(pow(0,-1))) echo "1/0 est indéfini<br />";
        else echo "1/0 n'est pas indéfini<br />";
    if (is_nan(sqrt(-1))) echo "racine carrée de -1 est indéfini<br />";
        else echo "racine carrée de -1 n'est pas indéfini<br />";
?>
```

Cela retournera :

```
3 est fini
1/0 n'est pas fini
racine carrée de -1 n'est pas fini
3 n'est pas infini
1/0 est infini
racine carrée de -1 n'est pas infini
3 n'est pas indéfini
1/0 n'est pas indéfini
racine carrée de -1 est indéfini
```

Avec les opérateurs

Dans les versions actuelles de PHP, cela ne s'applique qu'aux valeurs retournées par les fonctions et non par les opérateurs. Ainsi 1/0 retournera `FALSE` *accompagné d'un message d'erreur, or* `FALSE` *converti en entier donne 0, qui est fini. Vous aurez donc la surprise de constater que* `is_finite(1/0)` *retourne* `TRUE` *et non* `FALSE` *comme attendu.*

Trigonométriques

pi()

Cette fonction retourne une valeur approximative de Pi. 3.1415926535898 (revient à utiliser `M_PI`).

Syntaxe	`double pi(void)`
`retour`	Valeur approximative de Pi.

cos()

Retourne la valeur du cosinus de l'angle exprimé en radians.

Syntaxe	`double cos(double $angle)`
`$angle`	Angle exprimé en radians.
`retour`	Cosinus de l'angle.

acos()

Retourne l'arc cosinus de la valeur donnée (l'arc cosinus étant l'angle pour lequel le cosinus vaut `$cos`).

Syntaxe	`double acos(double $cos)`
`$cos`	Valeur sans unité.
`retour`	Angle en radians.

Voici un exemple de script utilisant les fonctions `cos()`, `acos()` et `pi()`, ainsi que les constantes `M_PI` et `M_SQRT1_2` :

```
<?php
    echo "cos(M_PI):".cos(M_PI)."<br />";
    echo "cos(pi()):".cos(pi())."<br />";
    echo "cos(M_PI/2):".cos(M_PI/2)."<br />";
    echo "cos(3*M_PI/2):".cos(3*M_PI/2)."<br />";
    echo "cos(M_PI/4):".cos(M_PI/4)."<br />";
    echo "cos(3*M_PI/4):".cos(3*M_PI/4)."<br />";
    echo "cos(-M_PI/4):".cos(-M_PI/4)."<br />";
    echo "cos(-3*M_PI/4):".cos(-3*M_PI/4)."<br />";
    echo "acos(M_SQRT1_2):".acos(M_SQRT1_2)."<br />";
    echo "acos(-M_SQRT1_2):".(-M_SQRT1_2)."<br />";
?>
```

Voici le résultat obtenu :

```
cos(M_PI):-1
cos(pi()):-1
cos(M_PI/2):6.1230317691119E-017
cos(3*M_PI/2):-1.8369095307336E-016
cos(M_PI/4):0.70710678118655
cos(3*M_PI/4):-0.70710678118655
cos(-M_PI/4):0.70710678118655
cos(-3*M_PI/4):-0.70710678118655
acos(M_SQRT1_2):0.78539816339745
acos(-M_SQRT1_2):-0.78539816339745
```

`M_PI` et `pi()` retournent tous les deux la même valeur. Les valeurs de `cos(Pi/2)` et `cos(3*Pi/2)` retournent un résultat proche de 0, mais pas 0. En effet, la valeur de Pi étant approximative, celle du cosinus est affectée.

`Cos(Pi/4)` et `cos(-Pi/4)` retournent la moitié de la racine carrée de 2. `Cos(3*Pi/4)` et `cos(-3*Pi/4)` sont opposés comme prévu.

`acos`(racine carrée de 2 sur 2) retourne `Pi/4` et `acos`(-racine carrée de 2 sur 2) retourne `-Pi/4`.

sin()

Retourne la valeur du sinus de l'angle exprimé en radians.

Syntaxe	`double sin(double $angle)`
`$angle`	Angle en radians.
`retour`	Sinus de l'angle.

asin()

Retourne l'arc sinus de la valeur donnée (l'arc cosinus étant l'angle pour lequel le sinus vaut `$sin`).

Syntaxe	`double asin(double $sin)`
`$sin`	Valeur sans unité.
`retour`	Angle en radians.

tan()

Retourne la valeur de la tangente de l'angle exprimée en radians.

Syntaxe	`double tan(double $angle)`
`$angle`	Angle en radians.
`retour`	Tangente de l'angle.

atan()

Retourne l'arc tangente de la valeur donnée (l'arc tangente étant l'angle pour lequel la tangente vaut `$tan`).

Syntaxe	`double atan(double $tan)`
`$tan`	Valeur sans unité.
`retour`	Angle en radians compris entre `-PI/2` et `PI/2`.

atan2()

Retourne l'arc tangente de `$sin/$cos` en tenant compte de leurs signes afin de déterminer de façon plus précise l'angle.

Syntaxe	`double atan2(double $sin, double $cos)`
`$sin`	Valeur sans unité.
`$cos`	Valeur sans unité.
`retour`	Angle en radians compris entre `-PI` et `PI` inclus. `[-PI;PI]`.

deg2rad()

Retourne en radians un angle exprimé en degrés.

Syntaxe	`double deg2rad(double $angle)`
`$angle`	Angle en degrés.
`retour`	Angle en radians.

rad2deg()

Retourne en degrés un angle exprimé en radians.

Syntaxe	`double rad2deg(double $angle)`
`$angle`	Angle en radians.
`retour`	Angle en degrés.

Logarithmiques

exp()

Retourne l'exponentiel de la valeur donnée.

Syntaxe	`double exp(double $reel)`
`$reel`	Valeur réelle.
`retour`	Exponentiel `$reel`.

log()

Retourne le logarithme népérien de la valeur donnée.

Syntaxe	`double log(double $reel)`
`$reel`	Valeur réelle dont on veut connaître le logarithme népérien.
`retour`	Valeur réelle du logarithme népérien de `$reel`.

log10()

Retourne le logarithme décimal.

Syntaxe	`double log10(double $reel)`
`$reel`	Valeur réelle dont on veut connaître le logarithme décimal.
`retour`	Logarithme décimal de `$reel`.

cosh()

Retourne la valeur du cosinus hyperbolique de l'angle, autrement dit `(exp($angle) + exp(-$angle))/2`.

Syntaxe	`double cosh(double $angle)`
`$angle`	Angle hyperbolique.
`retour`	Cosinus hyperbolique de l'angle.

acosh()

Retourne l'arc cosinus hyperbolique de la valeur donnée (l'arc cosinus hyperbolique étant l'angle pour lequel le cosinus hyperbolique vaut `$cosh`).

Syntaxe	`double acosh(double $cosh)`
`$cosh`	Valeur sans unité.
`retour`	Angle hyperbolique.

sinh()

Retourne la valeur du sinus hyperbolique de l'angle, autrement dit `(exp($angle) - exp(-$angle))/2`.

Syntaxe	`double sinh(double $angle)`
`$angle`	Angle hyperbolique.
`retour`	Sinus hyperbolique de l'angle.

asinh()

Retourne l'arc sinus hyperbolique de la valeur donnée (l'arc sinus hyperbolique étant l'angle hyperbolique pour lequel le sinus hyperbolique vaut `$sinh`).

Syntaxe	`double asinh(double $sinh)`
`$sinh`	Valeur sans unité.
`retour`	Angle hyperbolique.

tanh()

Retourne la valeur de la tangente hyperbolique de l'angle.

Syntaxe	`double tanh(double $angle)`
`$angle`	Angle hyperbolique.
`retour`	Tangente hyperbolique de l'angle.

atanh() (non disponible sous Windows)

Retourne l'arc tangente hyperbolique de l'angle donné (l'arc tangente hyperbolique étant l'angle hyperbolique pour lequel la tangente hyperbolique vaut `$tanh`).

Syntaxe	`double atanh(double $tanh)`
`$tanh`	Valeur sans unité.
`retour`	Angle hyperbolique.

Puissance

pow()

Cette fonction retourne `$base` à la puissance `$exposant`.

Syntaxe	`numeric pow(numeric $base, numeric $exposant)`
`$base`	Nombre à élever.
`$exposant`	Puissance.
`retour`	Entier si possible, sinon réel pouvant valoir `NAN` si la puissance ne peut être calculée (ex. : racine carrée d'un nombre négatif).

Voici un exemple d'utilisation de la fonction `pow()` :

Listing 6.2 : pow.php

```
<?php
    // exemple simple: 2^3
    echo pow(2, 3);
    echo "<br />";
    // exemple simple: (-1)^10
    echo pow(-1, 10);
    echo "<br />";
    // exemple d'erreur: racine de -4  (-4)^0.5
    echo pow(-4, 0.5);
?>
```

En sortie, on obtiendra :

```
8
1
NAN
```

sqrt()

Cette fonction retourne la racine carrée de la valeur entrée en paramètre.

Syntaxe	`double sqrt(double $reel)`
`$reel`	Valeur dont on veut connaître la racine carrée.
`retour`	Racine carrée de `$reel`.

Voici un exemple d'utilisation de la fonction `sqrt()` :

Listing 6.3 : sqrt.php

```
<?php
    echo sqrt(4)."<br />";
    echo sqrt(9)."<br />";
    echo sqrt(5)."<br />";
    echo sqrt(-3)."<br />";
?>
```

En sortie, on obtiendra :

```
2
3
2.2360679774998
NAN
```

Distance entre deux points
Pour calculer la distance entre le point A de coordonnées (x1,y1) et B de coordonnées (x2,y2), il suffit de faire `sqrt(pow(x2-x1,2)+pow(y2-y1,2))`.

Arrondi

ceil()

Arrondit à l'entier supérieur.

Cette fonction retourne l'entier directement supérieur. L'entier retourné est malgré tout considéré de type `double`, car ce type permet de traiter des nombres plus grands que les `int`.

Syntaxe	`double ceil(double $reel)`
`$reel`	Valeur à arrondir.
`retour`	Valeur arrondie mais de type réel.

Voici un exemple d'utilisation de la fonction `ceil()` :

Listing 6.4 : ceil.php

```
<?php
    echo ceil(3.0)."<br />";
    echo ceil(3.1)."<br />";
    echo ceil(3.2)."<br />";
    echo ceil(3.3)."<br />";
    echo ceil(3.4)."<br />";
    echo ceil(3.5)."<br />";
    echo ceil(3.6)."<br />";
    echo ceil(3.7)."<br />";
    echo ceil(3.8)."<br />";
    echo ceil(3.9)."<br />";
?>
```

En sortie, on obtiendra :

```
3
4
4
4
4
4
4
4
4
4
```


floor()

Arrondit à l'entier inférieur.

Cette fonction retourne l'entier directement inférieur. L'entier retourné est malgré tout considéré de type `double`, car ce type permet de traiter des nombres plus grands que les `int`.

Syntaxe	`double floor(double $reel)`
`$reel`	Valeur à arrondir.
`retour`	Valeur arrondie mais de type réel.

Voici un exemple d'utilisation de la fonction `floor()` :

Listing 6.5 : floor.php

```
<?php
    echo floor (3.0)."<br />";
    echo floor (3.1)."<br />";
    echo floor (3.2)."<br />";
```

```
    echo floor (3.3)."<br />";
    echo floor (3.4)."<br />";
    echo floor (3.5)."<br />";
    echo floor (3.6)."<br />";
    echo floor (3.7)."<br />";
    echo floor (3.8)."<br />";
    echo floor (3.9)."<br />";
?>
```

En sortie, on obtiendra :

3
3
3
3
3
3
3
3
3
3

round()

Arrondit selon les règles d'usage en mathématiques.

S'il n'y a qu'un argument, cette fonction retourne l'entier le plus proche. S'il y a deux arguments elle retourne la valeur arrondie selon la précision donnée en second argument.

Syntaxe	`double round(double $reel [, int $precision])`
`$reel`	Valeur à arrondir.
`$precision`	Précision souhaitée (nombre de chiffres après la virgule).
`retour`	Nombre arrondi selon la précision désirée.

Voici un exemple d'utilisation de la fonction `round()` :

Listing 6.6 : round.php

```
<?php
    echo round (3.0)."<br />";
    echo round (3.1)."<br />";
    echo round (3.2)."<br />";
    echo round (3.3)."<br />";
    echo round (3.4)."<br />";
    echo round (3.5)."<br />";
    echo round (3.6)."<br />";
    echo round (3.7)."<br />";
    echo round (3.8)."<br />";
    echo round (3.9)."<br />";
```

```
    echo round (3.123456789,1)."<br />";
    echo round (3.123456789,2)."<br />";
    echo round (3.123456789,3)."<br />";
    echo round (3.123456789,4)."<br />";
    echo round (3.123456789,5)."<br />";
    echo round (3.123456789,6)."<br />";
    echo round (3.123456789,7)."<br />";
    echo round (3.123456789,8)."<br />";
    echo round (3.123456789,9)."<br />";
?>
```

En sortie, on obtiendra :

```
3
3
3
3
3
4
4
4
4
4
3.1
3.12
3.123
3.1235
3.12346
3.123457
3.1234568
3.12345679
3.123456789
```

En effet, quand le chiffre de rang inférieur est supérieur ou égal à cinq, la règle est d'arrondir au chiffre supérieur.

Hasard

Nombre pseudo-aléatoire
L'ordinateur ne connaît pas le hasard. Mais, comme il est parfois utile de générer un nombre aléatoirement, il a fallu trouver une astuce pour faire comme si l'ordinateur en était capable. Pour arriver à ce résultat, il a fallu générer une suite (la plus longue possible) de nombres répondant à des critères statistiques précis. Lorsque l'on a besoin d'un nombre aléatoire, le système va piocher un nombre dans cette liste (le suivant dans la liste). Mais, si l'on n'y prend garde, après chaque redémarrage du logiciel, le nombre retourné risque d'être toujours le premier élément de la liste. Pour pallier ce problème, il faudrait donc choisir aléatoirement (on se mord la queue) l'endroit où commencer dans la liste. Pour choisir de façon plus ou moins aléatoire


REMARQUE

cet endroit, il est généralement fait appel à l'heure (les microsecondes). Comme tout ceci n'est pas à proprement parler du hasard, il est plus rigoureux de parler de nombre pseudo-aléatoire que de nombre aléatoire.

PHP propose deux types de fonctions : les classiques et les fonctions préfixées par `mt_` (utilisant un autre algorithme).

Fonctions "classiques"

srand()

Initialise les générateurs de nombres aléatoires.

Attention de ne pas utiliser cette fonction avant tous les appels dans une boucle par exemple. Il est préférable de ne l'utiliser qu'une fois avant la génération de plusieurs valeurs aléatoires.

Syntaxe	`void srand([int $seed])`
`$seed`	Une valeur quelconque qui doit changer d'une fois sur l'autre. On a pour habitude de prendre le nombre de microsecondes depuis la dernière seconde entière. `srand((double)microtime()*1000000).`

rand()

Générateur de nombres aléatoires.

Sans arguments, elle renvoie un nombre entre 0 et la valeur maximale que peut retourner la fonction (i.e. `getRandMax()`). Sinon, elle renvoie un nombre compris entre les deux valeurs (incluses) fournies.

Depuis PHP 4.2.0, il n'est plus nécessaire de faire appel au préalable à `srand()`, puisque cet appel est effectué automatiquement s'il n'a pas déjà eu lieu.

Syntaxe	`int rand([int $min, int $max])`
`$min`	Plus petite valeur que doit retourner la fonction.
`$max`	Plus grande valeur que doit retourner la fonction.
`retour`	Un entier entre minimum et maximum inclus.

getRandMax()

Retourne la plus grande valeur pouvant être atteinte par la fonction `rand()`.

Syntaxe	`int getRandMax(void)`
`retour`	Plus grande valeur que peut retourner la fonction `rand()`.

Voici un exemple utilisant les fonctions `getRandMax()`, `srand()` et `rand()`.

Listing 6.7 : rand.php

```
<?php
    echo "getRandMax()=".getRandMax()."<br />";
    srand((double)microtime()*1000000);
    echo rand()."<br />";
    echo rand()."<br />";
    echo rand()."<br />";
    echo rand()."<br />";
    echo rand(0, 2)."<br />";
    echo rand(0, 2)."<br />";
    echo rand(0, 2)."<br />";
    echo rand(0, 2)."<br />";
    echo rand(0, 2)."<br />";
    echo rand(0, 2)."<br />";
    echo rand(0, 2)."<br />";
    echo rand(0, 2)."<br />";
?>
```

Le résultat obtenu est le suivant (bien entendu, à part la première ligne, il varie d'une exécution à l'autre) :

```
getRandMax()=32767
1282
31943
31613
7866
0
1
2
0
2
1
0
2
```


Fonctions "mt"

mt_srand()

Initialise le générateur de nombres aléatoires.

Attention de ne pas utiliser cette fonction avant tous les appels dans une boucle par exemple. Il est préférable de ne l'utiliser qu'une fois avant la génération de plusieurs valeurs aléatoires.

Syntaxe	`void mt_srand([int $seed])`
`$seed`	Valeur quelconque qui doit changer d'une fois sur l'autre. On a pour habitude de prendre le nombre de microsecondes depuis la dernière seconde entière. `mt_srand((double)microtime()*1000000).`

mt_rand()

Générateur de nombres aléatoires.

`mt_rand` est une alternative à la fonction `rand`. Elle est bien plus rapide (environ quatre fois) et utilise la méthode de Mersenne Twister.

Sans arguments, elle renvoie un nombre entre 0 et la valeur maximale que peut retourner la fonction (i.e. `mt_getRandMax()`). Sinon, elle renvoie un nombre compris entre les deux valeurs fournies incluses.

Depuis PHP 4.2.0, il n'est plus nécessaire de faire appel au préalable à `mt_srand()`, puisque cet appel est effectué automatiquement s'il n'a pas déjà eu lieu.

Syntaxe	`int mt_rand([int $min, int $max])`
`$min`	Plus petite valeur que doit retourner la fonction.
`$max`	Plus grande valeur que doit retourner la fonction.
`retour`	Entier entre minimum et maximum inclus.

mt_getRandMax()

Retourne la plus grande valeur pouvant être atteinte par la fonction `mt_rand()`.

Syntaxe	`int mt_getRandMax(void)`
`retour`	Plus grande valeur que peut retourner `mt_rand()`.

Voici un exemple utilisant les fonctions `mt_getRandMax()`, `mt_srand()` et `mt_rand()` :

Listing 6.8 : mt_rand.php

```
<?php
    echo "mt_getRandMax()=".mt_getRandMax()."<br />";
    mt_srand((double)microtime()*1000000);
    echo mt_rand()."<br />";
    echo mt_rand()."<br />";
    echo mt_rand()."<br />";
    echo mt_rand()."<br />";
    echo mt_rand(0,2)."<br />";
    echo mt_rand(0,2)."<br />";
    echo mt_rand(0,2)."<br />";
    echo mt_rand(0,2)."<br />";
    echo mt_rand(0,2)."<br />";
    echo mt_rand(0,2)."<br />";
    echo mt_rand(0,2)."<br />";
    echo mt_rand(0,2)."<br />";
?>
```

Le résultat obtenu est le suivant (bien entendu, à part la première ligne, il varie d'une exécution à l'autre).

```
mt_getRandMax()=2147483647
617390894
505571108
175954693
741777589
2
0
1
1
2
0
2
0
```

Autres

lcg_value()

Générateur de congruence combinée linéaire.

Cette fonction retourne un nombre pseudo-aléatoire compris entre 0 et 1. Elle combine deux nombres de période 2 ^ 31-85 et 2 ^ 31-249. La période de cette fonction est égale au produit de ces deux nombres premiers soit (2 ^ 31-85)*(2 ^ 31-249).

Syntaxe	`double lcg_value()`
`retour`	Réel pseudo-aléatoire compris entre 0 et 1.

Voici un exemple utilisant la fonction `lcg_value()` :

Listing 6.9 : lcg.php

```
<?php
    echo(lcg_value()."<br />");
    echo(lcg_value()."<br />");
    echo(lcg_value()."<br />");
    echo(lcg_value()."<br />");
    echo(lcg_value()."<br />");
    echo(lcg_value()."<br />");
    $a = 3;
    $b = 7;
    // Voici une formule pour obtenir un nombre entre $a et $b inclus
    echo(floor($a + lcg_value()*($b-$a+1))."<br />");
    echo(floor($a + lcg_value()*($b-$a+1))."<br />");
    echo(floor($a + lcg_value()*($b-$a+1))."<br />");
    echo(floor($a + lcg_value()*($b-$a+1))."<br />");
?>
```

Voici le résultat retourné :

```
0.30513436806404
0.068577150355096
0.57782412638032
0.9158474922213
0.0046724566600712
0.1675198682609
7
3
4
6
```

Conversion de bases

base_convert()

Retourne un nombre converti d'une base dans une autre.

Syntaxe	`string base_convert(string $nombre, int $depuisBase, int $versBase)`
`$nombre`	Nombre à convertir écrit en base `$depuisBase`.
`$depuisBase`	Base d'origine entre 2 et 36.
`$versBase`	Base d'arrivée entre 2 et 36.
`retour`	Nombre en base `$versBase`.

binDec()

Retourne la conversion d'un nombre binaire inférieur à 2147483647 (31 bits à 1) en nombre de base 10 (décimal).

Syntaxe	`int binDec(string $binaire)`
`$binaire`	Nombre binaire de moins de 32 bits.
`retour`	Nombre en base 10.

decBin()

Retourne la conversion d'un nombre décimal (i.e. base 10) inférieur à 2147483647 (31 bits à 1) en nombre de base 2 (binaire).

Syntaxe	`string decBin(int $decimal)`
`$decimal`	Nombre décimal (i.e. base 10) inférieur à 2147483647.
`retour`	Nombre en base 2.

decHex()

Retourne la conversion d'un nombre décimal (i.e. base 10) inférieur à 2147483647 (31 bits à 1 ou encore 7FFFFFFF en hexadécimal) en nombre de base 16.

Syntaxe	`string decHex(int $decimal)`
`$decimal`	Nombre décimal (i.e. base 10) inférieur à 2147483647.
`retour`	Nombre en base 16.

hexDec()

Retourne la conversion d'un nombre hexadécimal inférieur à 7FFFFFFF (31 bits à 1 ou encore 2147483647 en décimal) en nombre décimal (i.e. base 10).

Syntaxe	`int hexDec(string $hexadecimal)`
`$hexadecimal`	Nombre décimal inférieur à 7FFFFFFF.
`retour`	Nombre décimal (i.e. base 10).

decOct()

Retourne la conversion d'un nombre décimal (i.e. base 10) inférieur à 2147483647 (31 bits à 1 ou encore 17777777777 en octal) en nombre de base 8.

Syntaxe	`string decOct(int $decimal)`
`$decimal`	Nombre décimal (i.e. base 10) inférieur à 2147483647.
`retour`	Nombre en base 8.

octDec()

Retourne la conversion d'un nombre octal inférieur à 17777777777 (31 bits à 1 ou encore 2147483647 en décimal) en nombre décimal (i.e. base 10).

Syntaxe	`int octDec(string $octal)`
`$octal`	Nombre octal inférieur à 17777777777.
`retour`	Nombre décimal (i.e. base 10).

Voici un exemple utilisant toutes ces fonctions de conversion de bases.

Listing 6.10 : bases.php

```
<?php
    echo base_convert("28", 10, 2)."<br />";  // 11100
    echo base_convert("28", 10, 8)."<br />";  // 34
    echo base_convert("28", 10, 16)."<br />"; // 1c
    echo decBin(28)."<br />";                 // 11100
    echo binDec("11100")."<br />";            // 28
    echo decOct(28)."<br />";                 // 34
    echo octDec("34")."<br />";               // 28
    echo decHex(28)."<br />";                 // 1c
    echo hexDec("1c")."<br />";               // 28
?>
```

Et le résultat retourné est :

```
11100
34
1c
11100
28
34
28
1c
28
```


Autres

abs()

Cette fonction retourne la valeur absolue de l'argument. Si l'argument est un réel, la valeur retournée est un réel ; sinon, c'est un entier.

Syntaxe	`mixed abs(mixed $nombre)`
`$nombre`	Valeur réelle ou entière.
`retour`	Valeur absolue du nombre.

max()

Retourne le plus grand élément.

Syntaxe	`mixed max(mixed $arg1 [, mixed $arg2, ...[, mixed $argn]])`
`$arg1`	Un tableau, une chaîne de caractères, un réel ou un entier.

`$arg2`	Une chaîne de caractères, un réel ou un entier si `$arg1` n'est pas un tableau.
`$argn`	Une chaîne de caractères, un réel ou un entier si `$arg1` n'est pas un tableau.
`retour`	Le plus grand élément du tableau si l'argument est un tableau. Si tous les arguments sont de type `string`, la valeur retournée est la dernière valeur par rapport à l'ordre alphabétique. Si tous les arguments numériques sont de type `int`, c'est un entier qui est retourné ; sinon c'est un réel.

min()

Retourne le plus petit élément.

Syntaxe	`mixed min(mixed $arg1 [, mixed $arg2, ... [, mixed $argn])`
`$arg1`	Un tableau, une chaîne de caractères, un réel ou un entier.
`$arg2`	Une chaîne de caractères, un réel ou un entier si `$arg1` n'est pas un tableau.
`$argn`	Une chaîne de caractères, un réel ou un entier si `$arg1` n'est pas un tableau.
`retour`	Le plus petit élément du tableau si l'argument est un tableau. Si tous les arguments sont de type `string`, la valeur retournée est la première valeur par rapport à l'ordre alphabétique. Si tous les arguments numériques sont de type `int`, c'est un entier qui est retourné ; sinon c'est un réel.

number_format()

Formate un nombre.

Cette fonction prend un, deux ou quatre paramètres.

S'il n'y a qu'un paramètre, il sera formaté sans décimal et avec une virgule séparant les milliers.

S'il y a deux paramètres, le deuxième argument est le nombre de décimales à afficher. Elles sont alors séparées par une virgule.

S'il y a quatre paramètres, les deux premiers sont les mêmes que dans le cas de deux paramètres, les suivants sont les séparateurs de décimales et de milliers.

Syntaxe	`string number_format(double $nombre [, int $decimales [, string $delimiteur , string $milliers]])`
`$nombre`	Nombre à formater.
`$decimales`	Nombre de décimales à afficher.
`$delimiteur`	Séparateur de décimales (', ' en français, '.' en anglais).
`$milliers`	Séparateur de milliers (' ' en français, ', ' en anglais).
`retour`	Chaîne de caractères du nombre formaté.

Voici un exemple d'utilisation de la fonction `number_format()` :

Listing 6.11 : numberf.php

```
<?php
    $number = 123456789.12345;
    // exemple avec un seul argument
    echo number_format($number);
    echo "<br />";
    // exemple avec deux arguments
    echo number_format($number, 3);
    echo "<br />";
    // exemple de notation anglaise avec 4 arguments
    echo number_format($number, 3, '.', ',');
    echo "<br />";
    // exemple de notation française avec 4 arguments
    echo number_format($number, 3, ',', ' ');
?>
```

En sortie, on obtiendra :

```
123,456,789
123,456,789.123
123,456,789.123
123 456 789,123
```


6.2. Calculs de précision

Les calculs de précision utilisent la bibliothèque `BCMath`.

Installation

Sous Windows

La version distribuée par PHP inclut le support de la bibliothèque `BCMath` (c'est également vrai pour le kit EasyPHP). Vous n'aurez donc rien à faire de particulier pour en profiter.

Sous Linux

Assurez-vous d'avoir compilé PHP avec l'option `--enable-bcmath`.


RENVOI

Vous pouvez vous reporter au chapitre "Installation" pour plus de détails sur la compilation de PHP.

Vérification

Vous pouvez vous assurer que la bibliothèque `BCMath` est bien disponible par un simple script contenant `<?php phpinfo(); ?>` et qui doit laisser apparaître :

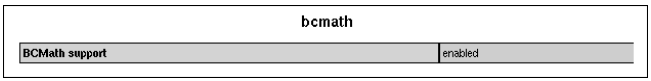


Figure 6.1 : *phpinfo()*

Utilisation

Avec la bibliothèque `BCMath`, les nombres sont représentés par des chaînes de caractères. Ils ne sont donc pas a priori limités en taille et, par conséquent, n'ont pas de limite théorique en précision.

bcscale()

Définit la précision par défaut (remplace le paramètre `bcmath.scale` du fichier *php.ini*).

Syntaxe	`boolean bcscale(int $precision)`
`$precision`	Entier exprimant le nombre de décimales par défaut.
`retour`	`TRUE`.

bcadd()

Ajoute deux nombres.

Syntaxe	`string bcadd(string $operande1, string $operande2 [, int $precision])`
`$operande1`	Chaîne de caractères représentant un nombre.
`$operande2`	Chaîne de caractères représentant un nombre.
`$precision`	Nombre de décimales.
`retour`	Chaîne de caractères représentant `$operande1+$operande2`.

bcsub()

Soustrait deux nombres.

Syntaxe	`string bcsub(string $operande1, string $operande2 [, int $precision])`
`$operande1`	Chaîne de caractères représentant un nombre.
`$operande2`	Chaîne de caractères représentant un nombre.
`$precision`	Nombre de décimales.
`retour`	Chaîne de caractères représentant `$operande1-$operande2`.

bcmul()

Multiplie deux nombres.

Syntaxe	`string bcmul(string $operande1, string $operande2 [, int $precision])`
`$operande1`	Chaîne de caractères représentant un nombre.
`$operande2`	Chaîne de caractères représentant un nombre.
`$precision`	Nombre de décimales.
`retour`	Chaîne de caractères représentant `$operande1*$operande2`.

bcdiv()

Divise deux nombres.

Syntaxe	`string bcdiv(string $operande1, string $operande2 [, int $precision])`
`$operande1`	Chaîne de caractères représentant un nombre.
`$operande2`	Chaîne de caractères représentant un nombre.
`$precision`	Nombre de décimales.
`retour`	Chaîne de caractères représentant `$operande1/$operande2`.

bcmod()

Retourne le reste d'une division.

Syntaxe	`string bcmod(string $operande1, string $operande2)`
`$operande1`	Chaîne de caractères représentant un nombre.
`$operande2`	Chaîne de caractères représentant un nombre.
`retour`	Chaîne de caractères représentant `$operande1 % $operande2`.

bcpow()

Élève un nombre à une puissance.

Syntaxe	`string bcpow(string $operande1, string $operande2 [, int $precision])`
`$operande1`	Chaîne de caractères représentant un nombre.
`$operande2`	Chaîne de caractères représentant un nombre.

`$precision`	Nombre de décimales.
`retour`	Chaîne de caractères représentant `$operande` à la puissance `$operande2`.

bcsqrt()

Retourne la racine carrée d'un nombre.

Syntaxe	`string bcsqrt(string $operande [, int $precision])`
`$operande`	Chaîne de caractères représentant un nombre.
`$precision`	Nombre de décimales.
`retour`	Chaîne de caractères représentant la racine carrée de l'opérande.

bccomp()

Compare deux nombres.

Syntaxe	`int bccomp(string $operande1, string $operande2 [, int $precision])`
`$operande1`	Chaîne de caractères représentant un nombre.
`$operande2`	Chaîne de caractères représentant un nombre.
`$precision`	Nombre de décimales.
`retour`	`0` si les deux nombres sont identiques, `1` si `$operande1` est plus grand que `$operande2`, sinon `-1`.

Voici un exemple de calcul utilisant la bibliothèque `BCMath`.

Listing 6.12 : precision.php

```
<?php
    bcscale(10);
    $a = "12345678901234567890";
    $b = "12345678";
    echo "Addition:<br />";
    echo ($a + $b)."<br />";
    echo bcadd($a, $b)."<br />";
    echo "Soustraction:<br />";
    echo ($a - $b)."<br />";
    echo bcsub($a, $b)."<br />";
    echo "Multiplication:<br />";
    echo ($a * $b)."<br />";
    echo bcmul($a , $b)."<br />";
    echo "Division:<br />";
```

```
    echo ($a / $b)."<br />";
    echo bcdiv($a, $b)."<br />";
    echo "Restant de la division:<br />";
    echo ($a % $b)."<br />";
    echo bcmod($a, $b)."<br />";
    echo "Exposant:<br />";
    echo pow($a, 2)."<br />";
    echo bcpow($a, 2)."<br />";
    echo "Racine carrée:<br />";
    echo sqrt($a)."<br />";
    echo bcsqrt($a)."<br />";
?>
```

Et le résultat retourné est le suivant :

```
Addition:
1.23456789012E+19
12345678901246913568.0000000000
Soustraction:
1.23456789012E+19
12345678901222222212.0000000000
Multiplication:
1.52415776406E+26
152415776406035777639079420
Division:
1000000073E+12
1000000073000.0059850904
Restant de la division:
11681353
73890
Exposant:
Warning: Invalid argument(s) passed to pow() in precision.php on line 21
152415787532388367501905199875019052100
Racine carrée:
3513641828.82
3513641828.8201442530
```

On peut remarquer l'erreur consécutive à un calcul du reste de la division sans prendre garde aux limites de précision de PHP ; qui plus est, PHP refuse de calculer la puissance si l'on ne passe pas par `BCMath`.


REMARQUE

Affichage des nombres
Pour une valeur donnée, le nombre de chiffres affichés dépend de la configuration de PHP et, en particulier, du paramètre `precision` *du fichier php.ini. Cette remarque ne s'applique évidemment pas aux valeurs retournées par* `BCMath`*, qui sont, de toute façon, des chaînes de caractères.*

Chapitre 7

La manipulation des chaînes de caractères

7.1. Généralités

Ce chapitre est peut-être le plus important, l'utilisation des chaînes de caractères étant en effet inévitable pour produire le code HTML désiré.

Les chaînes de caractères peuvent être encadrées soit par des apostrophes (') soit par des guillemets (").

Lorsque vous utilisez des guillemets, vous pouvez intégrer directement des noms de variables au sein de la chaîne de caractères, et celles-ci seront subtituées par leurs valeurs. Ce n'est pas le cas si vous utilisez les apostrophes. Ainsi,

```
<?php
    $monNom = "Emma";
    echo "Bonjour $monNom <br />";
    echo 'Bonjour $monNom <br />';
?>
```

affichera :

```
Bonjour Emma
Bonjour $monNom
```

Dans le cas de l'utilisation des guillemets, nous aurons donc besoin de faire appel aux séquences d'échappement afin de pouvoir afficher le signe '$'. Mais ceci concerne également les caractères suivants :

Tableau 7.1 : Séquences d'échappement

Notation	Description
`\n`	Changement de ligne.
`\r`	Retour chariot.
`\t`	Tabulation.
`\\`	Pour afficher `\` (et éviter la confusion avec le caractère d'échappement).
`\$`	Pour afficher `$` (et éviter la confusion avec l'interprétation d'une variable).
`\"`	Pour afficher `"` (et éviter la confusion avec la fin de la chaîne de caractères).

La seule séquence d'échappement disponible lors de l'utilisation des apostrophes et la suivante : `\'`. Elle met fin à la confusion qu'il pourrait y avoir avec la déclaration de fin de chaîne.

Si l'on veut utiliser des guillemets à l'intérieur d'autres guillemets, il est évident que cela pose problème, comme dans l'exemple suivant :

```
"Coluche a dit "Plus on est de fous moins y'a de riz", pas faux..."
```

L'analyseur syntaxique de PHP va reconnaître une première chaîne `"Coluche a dit "` puis des lettres qu'il ne saura pas interpréter `Plus on est de fous moins y'a de riz`, et, enfin, une chaîne de caractères `", pas faux..."`.

C'est pourquoi il faut préciser que le caractère " est un caractère à afficher. Pour cela, il a été décidé d'y ajouter \.

La phrase d'exemple s'écrit donc :

```
"Coluche a dit \"Plus on est de fous moins y'a de riz\", pas faux..."
```

Et, pour afficher \, il faut donc préciser que ce n'est pas le caractère d'échappement en le doublant : \\.


ATTENTION

Chemins de fichiers Windows
\\ est notamment à utiliser pour les chemins windows.
"*C:\monRepertoire\monFichier.php*" *s'écrit*
"*C:\\monRepertoire\\monFichier.php*".


REMARQUE

||
Pour écrire \\, il suffit de doubler chacun des caractères. On obtient donc \\\\ et ainsi de suite… Cette méthode permet de garder la liberté d'utiliser tous les caractères.

Le script suivant nous montre comment utiliser les séquences d'échappement :

Listing 7.1 : cc01.php

```
<?php
echo "1 - Facile";
echo "\n";
echo '2 - Facile';
echo "\n";
echo "3 - C'est plus dur";
echo "\n";
echo '4 - C\'est plus dur';
echo "\n";
$variable="valeur";
echo "5 - $variable";
echo "\n";
echo '6 - $variable';
echo "\n";
echo "7 - \$variable";
echo "\n";
echo "8 - citation \"PHP est facile\"";
echo "\n";
echo '9 - citation "PHP est facile"';
echo "\n";
// pour afficher \" avec les guillemets:
echo "10 - \\\"";
?>
```

Voici le résultat obtenu dans le code source généré :

```
1 - Facile
2 - Facile
3 - C'est plus dur
4 - C'est plus dur
5 - valeur
6 - $variable
7 - $variable
8 - citation "PHP est facile"
9 - citation "PHP est facile"
10 - \"
```

Pour accéder à des éléments d'une base de données, il faut particulièrement faire attention à l'utilisation des apostrophes. Par exemple :

```
"SELECT * FROM matable WHERE monchamp='$valeur'"
```

posera problème si `$valeur` contient une apostrophe. Heureusement, la fonction `addSlashes()` ajoute le caractère d'échappement devant les caractères problématiques. Il faut donc écrire :

```
"SELECT * FROM matable WHERE monchamp='".addSlashes($valeur)."'"
```

Le même genre de problème se retrouve lorsque l'on veut, par exemple, donner une valeur par défaut dans un champ HTML. Une fonction spécifique existe, qui s'appelle `htmlSpecialChars()`.

```
echo "<input type=\"hidden\" name=\"variable\"
      value=\"".htmlSpecialChars($maVariable)."\" />"
```

Par défaut, PHP est configuré avec l'option *magic_quotes_gpc* activée. Ainsi, les valeurs passées par les méthodes `GET`, `POST` et les cookies voient leurs apostrophes automatiquement précédées d'un anti-slash (il n'est alors plus nécessaire de faire appel à `addSlashes()` pour les utiliser dans des requêtes SQL). Il est possible de faire de même pour les fichiers et les bases de données, en utilisant le paramètre similaire `magic_quotes_runtime`.

Vous pouvez vous reporter au chapitre "Prise en main" pour plus de détails sur la configuration de PHP.

Rappelons que l'opérateur de concaténation est le point '.' et que l'opérateur '.=' est également valide. Ainsi `"Langage "."PHP"` vaut `"Langage PHP"`.

Afficher du texte

Il existe plusieurs façons d'afficher une chaîne de caractères. La manière la plus simple est l'utilisation de `echo`.

L'écriture sur plusieurs lignes peut se faire au moyen de <<< suivis d'un identifiant de fin de texte, qui ne sera pas dans le texte à écrire. L'identifiant placé à la fin du texte doit obligatoirement commencer sur la première colonne du fichier.

Voici différents cas d'utilisation de `echo` :

```
<?php
$variable = "valeur\n";
echo $variable;
echo "$variable";

echo "variable"," ",$variable;

echo "On peut
aussi écrire
sur plusieurs
lignes\n";

echo <<<END
On peut
aussi écrire
sur plusieurs
lignes de cette
façon\n
END;

echo <<<TEXTE
On peut
aussi écrire
sur plusieurs
lignes de cette
façon
TEXTE;
?>
```

Et voici le résultat de sortie :

```
valeur
valeur
variable valeur
On peut
aussi écrire
sur plusieurs
lignes
On peut
aussi écrire
sur plusieurs
lignes de cette
façon
On peut
aussi écrire
sur plusieurs
lignes de cette
façon
```

Une fonction très proche de la commande `echo` s'appelle `print()`. La différence est tellement infime que beaucoup se demandent où elle se trouve. `print()` est une fonction qui retourne un booléen, elle renvoie `TRUE` quand l'affichage s'est effectué.

`echo` est légèrement plus rapide que `print()`, dans la mesure où cette fonction ne retourne rien. Mais la différence reste minime.

Les deux fonctions `printf()` et `sprintf()` permettent respectivement d'afficher et de retourner une chaîne de caractères formatée.

printf()

Affiche une chaîne de caractères (éventuellement accompagnée de valeurs) selon un format donné.

Syntaxe	`void printf(string $format [, mixed $valeurs])`
`$format`	Chaîne décrivant le format attendu.
`$valeurs`	Les valeurs à formater.

sprintf()

Retourne une chaîne de caractères (éventuellement accompagnée de valeurs) selon un format donné.

Syntaxe	`string sprintf(string $format [, mixed $valeurs])`
`$format`	Chaîne décrivant le format attendu.
`$valeurs`	Les valeurs à formater.
`retour`	La chaîne formatée.

Les éléments formatés sont définis par le signe `%` suivi de différents caractères (ex. `:%03.2f`) dont voici une description :

- Le premier est facultatif. Il sert à déterminer le caractère qui complète les caractères manquants ; par défaut, il vaut le caractère d'espacement. Pour définir ce paramètre, il suffit de le faire précéder d'une apostrophe `'` (inutile pour le chiffre 0). (Cela peut, par exemple, permettre d'afficher un nombre sur trois chiffres, même s'il est inférieur à 100).
- Le deuxième paramètre est lui aussi optionnel. Il détermine si le résultat doit être aligné à droite ou à gauche. Par défaut, il sera aligné à droite ; pour le faire aligner à gauche, il suffit de placer le signe `-`.
- Le troisième paramètre est lui aussi optionnel. Il détermine le nombre de caractères minimum à retourner.

- Le quatrième paramètre est lui aussi optionnel. Il détermine le nombre de décimales à afficher pour les réels. Ce paramètre n'a bien sûr aucun effet sur les autres types ; il est précédé d'un point.
- Le cinquième paramètre est obligatoire. Il définit selon quel type l'argument doit être traité puis représenté. Voici les différents types possibles :

Tableau 7.2 : Les différents types

Lettre	Traité	Représenté
b	Entier	Binaire
c	Entier	Caractère avec sa valeur ASCII
d	Entier	Nombre décimal signé
u	Entier	Nombre décimal non signé
f	Réel	Réel
o	Entier	Nombre octal
s	Tel quel	Chaîne de caractères
x	Entier	Nombre hexadécimal
X	Entier	Nombre hexadécimal en majuscules

Écriture du caractère %
Pour écrire le caractère %, il suffit de le doubler afin d'éviter la confusion. Au moment de l'affichage un seul apparaîtra.

Voici quelques exemples utiles :

Listing 7.2 : sprintf.php

```
<?php
echo "Date<br /> ";
printf("%02d/%02d/%04d", 1, 7, 2002);
echo "<br />Pi avec 7 décimales<br />";
printf("%.7f", M_PI);
echo "<br />Ecriture d'une chaîne<br />";
printf(".:%s:.", "Salut !");
echo "<br />Ecriture d'une chaîne avec 20 caractères minimum<br />";
printf(".:%20s:.", "Salut");
echo "<br />Ecriture d'une chaîne avec 20 caractères minimum,".
     " aligné à gauche<br />";
printf(".:%-20s:.", "Salut");
echo "<br />Ecriture d'une chaîne avec 20 caractères minimum,".
     " complété par des -<br />";
printf(".:%'-20s:.", "Salut");
echo "<br />Ecriture d'une chaîne avec 20 caractères minimum,".
```

```
     " aligné à gauche, complété par des -<br />";
printf(".:%'--20s:.", "Salut");
echo "<br />Ecriture de %<br />";
printf("%%");
?>
```

Et voici le résultat obtenu :

```
Date
01/07/2002
Pi avec 7 décimales
3.1415927
Ecriture d'une chaîne
.:Salut !:.
Ecriture d'une chaîne avec 20 caractères minimum
.:               Salut:.
Ecriture d'une chaîne avec 20 caractères minimum, aligné à gauche
.:Salut :.
Ecriture d'une chaîne avec 20 caractères minimum, complété par des -
.:---------------Salut:.
Ecriture d'une chaîne avec 20 caractères minimum, aligné à gauche, complété par
✂ des -
.:Salut---------------:.
Ecriture de %
%
```

La fonction `vPrintf()` permet d'afficher un tableau. Elle fonctionne de la même manière que `printf()`.

vPrintf()

Affiche une chaîne de caractères formatée et construite à partir de données stockées dans un tableau.

Syntaxe	`boolean vPrintf(string $format, array $tableau)`
`$format`	Voir description de `printf()`.
`$tableau`	Tableau à afficher.
`retour`	Un booléen.

Voici un exemple :

```
<?php
$tableau=array("Element 1", "Element 2", "Element 3");
vPrintf("%2\$s|%1\$s|%3\$s", $tableau);
?>
```

dont le résultat obtenu est le suivant :

```
Element 2|Element 1|Element 3
```

vSPrintf()

Retourne une chaîne de caractères formatée et construite à partir de données stockées dans un tableau.

Syntaxe	`string vSPrintf(string $format, array $tableau)`
`$format`	Voir description de `vPrintf()`.
`$tableau`	Tableau à afficher.
`retour`	La chaîne de caractères.

sscanf()

Permet de faire l'analyse lexicale d'une chaîne de caractères (i.e. de récupérer les informations qui la composent).

Syntaxe	`mixed sscanf(string $chaine, string $format[, string &$variableSortie1 [, string &$variableSortie2[, …]]]`
`$chaine`	Chaîne de caractères dont on veut récupérer des éléments.
`$format`	Format de la chaîne à analyser.
`$variableSortieN`	Variables dans lesquelles vous souhaitez stocker les différents éléments extraits de la chaîne de caractères.
`retour`	Tableau avec les différentes parties reconnues de la chaîne de caractères, ou le nombre de sous-chaînes reconnues si des des variables sont passées par référence.

Un exemple d'utilisation de `sscanf()` :

```
<?php
    $phrase = "J'ai 25 ans et je suis Français";
    $tableau = sscanf($phrase, "J'ai %d ans et je suis %s");
    echo $tableau[0]."<br />\n";
    echo $tableau[1]."<br />\n";

    list($age, $nationalite) = sscanf($phrase, "J'ai %d ans et je suis %s");
    echo $age."<br />\n";
    echo $nationalite."<br />\n";

    $refage=0;
    $refnat="";
    echo sscanf($phrase, "J'ai %d ans et je suis %s", &$refage, &$refnat)
         ." éléments<br />\n";
    echo $refage."<br />\n";
    echo $refnat."<br />\n";
?>
```

Et le résultat obtenu :

```
25
Français
25
Français
2 éléments
25
Français
```

Manipuler les caractères

chr()

Retourne le caractère d'un code ASCII.

Syntaxe	`string chr(int $codeAscii)`
`$codeAscii`	Code ASCII du caractère à afficher.
`retour`	Caractère correspondant au code ASCII passé en paramètre.

ord()

Retourne le code ASCII du premier caractère d'une chaîne.

Syntaxe	`int ord(string $chaine)`
`$chaine`	Chaîne dont on veut le caractère ASCII du premier caractère.
`retour`	Code ASCII du premier caractère.

7.2. Fonctions de gestion des chaînes de caractères

Extraction et substitution

Extraction

substr()

Retourne une partie d'une chaîne de caractères.

Syntaxe	`string substr(string $chaine, int $debut [,int $nbCaracteres])`
`$chaine`	Chaîne de caractères de laquelle vous souhaitez extraire une partie.

`$debut`	Indice du caractère de départ. (Le premier caractère ayant l'indice 0, si la valeur est négative, le compte se fait depuis la droite, le dernier caractère ayant l'indice -1).
`retour`	La sous-chaîne demandée.

Listing 7.3 : substr.php

```
<?php
echo substr("abcdefghij", 2);
echo "<br />";
echo substr("abcdefghij", 0, 3);
echo "<br />";
echo substr("abcdefghij", 1, 4);
echo "<br />";
echo substr("abcdefghij", 5, 20);
echo "<br />";
echo substr("abcdefghij", -3, 2);
?>
```

Et le résultat obtenu est le suivant :

```
cdefghij
abc
bcde
fghij
hi
```

Parfois, nous ne sommes pas en mesure de connaître par avance la position recherchée dans une chaîne de caractères. Les fonctions suivantes ne nécessitent pas de connaître d'index.

La fonction `strstr()` permet de retrouver la première occurrence d'un caractère ou d'une chaîne de caractères dans une autre chaîne de caractères.

strstr()

Permet de récupérer la partie d'une chaîne de caractères située à partir de la première occurrence d'une sous-chaîne.

Syntaxe	`string strstr(string $chaine, string $souschaine)`
`$chaine`	Chaîne de caractères d'où vous souhaitez extraire une partie.
`$souschaine`	Chaîne de caractères définissant le début.
`retour`	Une sous-chaîne de `$chaine` comprenant `$souschaine` et la fin de `$chaine`.

Listing 7.4 : strstr.php

```
<?php
echo "strstr:".strstr("Voici un exemple stupide d'exemple", "exemple");
```

```
echo "\n";
echo "strstr:".strstr("thomas@toutestfacile.com", "@");
?>
```

En sortie, nous obtiendrons :

```
strstr:exemple stupide d'exemple
strstr:@toutestfacile.com
```


REMARQUE

strchr()

`strchr()` est un alias de `strstr()`, c'est-à-dire que ces deux fonctions sont en tous points identiques.

`strstr()` est une fonction sensible aux majuscules/minuscules. Pour ne pas prendre en compte la casse, il faut utiliser `stristr()`, qui fonctionne de la même façon.

Listing 7.5 : stristr.php

```
<?php
echo "strstr:".strstr("Voici un exemple stupide d'exemple", "Exemple");
echo "\n";
echo "stristr:".stristr("Voici un exemple stupide d'exemple", "Exemple");
?>
```

En sortie, nous obtiendrons :

```
strstr:
stristr:exemple stupide d'exemple
```

Dans le cas où nous avons utilisé `strstr()`, la sous-chaîne de caractères "`Exemple`" n'a pas été trouvée à cause de la majuscule.

Dans le cas où la sous-chaîne est présente plusieurs fois, il est possible de récupérer la partie située après la dernière occurrence à l'aide de `strrchr()`.

strrchr()

Permet de récupérer la partie d'une chaîne de caractères située à partir de la dernière occurrence d'une autre chaîne.

Syntaxe	`string strrchr(string $chaine, string $souschaine)`
`$chaine`	Chaîne de caractères d'où vous souhaitez extraire une partie.
`$souschaine`	Chaîne de caractères définissant le début.
`retour`	Une sous-chaîne de `$chaine` comprenant `$souschaine` et la fin de `$chaine`.

```
<?php
```

```
echo strrchr("Voici un exemple stupide d'exemple", "exemple");
echo "\n";
echo strrchr("article 1,article 2, article 3", ",");
?>
```

Et voici le résultat obtenu :

```
e
, article 3
```

Substitution

substr_replace()

Retourne une chaîne de caractères issue d'une chaîne dans laquelle une partie a été remplacée par une autre. Permet également d'insérer du texte.

Syntaxe	`string substr_replace(string $chaine, string $substitution, int $debut [, int $nbCaracteres])`
`$chaine`	Chaîne de caractères dans laquelle vous souhaitez remplacer du texte.
`$substitution`	Chaîne de caractères que vous souhaitez écrire à la place.
`$debut`	Indice du caractère de départ. (Le premier caractère ayant l'indice 0, si la valeur est négative, le compte se fait depuis la droite, le dernier caractère ayant l'indice -1).
`$nbCaracteres`	Nombre de caractères à remplacer.
`retour`	La chaîne de caractères modifiée.

Listing 7.6 : substrreplace.php

```
<?php
echo substr_replace("abCDEF", "AB", 0, 2);
echo "<br />";
echo substr_replace("100000 Euros", ",", -9, 0);
?>
```

```
ABCDEF
100,000 Euros
```

Il est possible d'utiliser cette fonction afin de réduire des chaînes de caractères. Par exemple, pour modifier "cette trop longue phrase" en "cette trop longue ph…", il suffit d'écrire :

```
substr_replace("cette trop longue phrase", "...", 20)
```

Il est assez rare que nous connaissions par avance la position des caractères dans la chaîne. Bien plus souvent, il s'agit d'un mot ou d'une expression que nous souhaitons remplacer par un(e) autre. Il est, par exemple, possible d'imaginer que l'on souhaite remplacer les mots vulgaires d'un forum par '(censuré)'. La fonction qui va nous le permettre s'appelle `str_replace()`.

str_replace()

Cette fonction permet de remplacer des occurrences par d'autres.

Syntaxe	`mixed str_replace(mixed $motif, mixed $remplacement, mixed $chaine [, int &$nombreRemplacements])`
`$motif`	Une chaîne de caractères ou un tableau de chaînes de caractères désignant le ou les éléments à rechercher.
`$remplacement`	Une chaîne de caractères ou un tableau de chaînes de caractères désignant le ou les éléments qui remplaceront les éléments de `$motif`.
`$chaine`	Une chaîne de caractères ou un tableau de chaînes de caractères désignant le ou les éléments à modifier.
`$nombreRemplacements`	Cette fonction ecrira dans la variable $nombre_remplacements, le nombre de remplacements effectués dans la chaîne de caractères. (Cette option n'existe que depuis la version 5.0.0 de PHP)
`retour`	La chaîne de caractères ou le tableau de chaînes de caractères modifié.

L'utilisation la plus simple consiste à remplacer une sous-chaîne par une autre dans une chaîne de caractères.

Voici un code source d'exemple :

Listing 7.7 : strreplace1.php

```
<?php
$phrase = "Jessicasse-croutes dans mon panier";
echo $phrase."<br />";
$phrase = str_replace("Jessica", "J'ai 6 ca", $phrase);
echo $phrase."<br />";
?>
```

Et le résultat :

```
Jessicasse-croutes dans mon panier
J'ai 6 casse-croutes dans mon panier
```

Si `$chaine` est un tableau, le remplacement s'effectue sur tous les éléments du tableau. En retour, nous obtenons alors un tableau des chaînes de caractères modifiées.

Voici un code source d'exemple :

Listing 7.8 : strreplace2.php

```
<?php
$phrases = array("Jessicasse-croutes dans mon panier",
                 "Jessicanapes dans mon salon",
                 "Jessicadeaux sous mon sapin");
echo $phrases[0]."<br />";
echo $phrases[1]."<br />";
echo $phrases[2]."<br />";
```

```
$phrases = str_replace("Jessica", "J'ai 6 ca", $phrases);
echo $phrases[0]."<br />";
echo $phrases[1]."<br />";
echo $phrases[2]."<br />";
?>
```

Et le résultat :

```
Jessicasse-croutes dans mon panier
Jessicanapes dans mon salon
Jessicadeaux sous mon sapin
J'ai 6 casse-croutes dans mon panier
J'ai 6 canapes dans mon salon
J'ai 6 cadeaux sous mon sapin
```

Il est également possible de tirer partie de cette fonction pour effectuer plusieurs modifications d'un seul coup, en utilisant des tableaux pour les variables `$motif` et `$remplacement`.

Voici un code source d'exemple :

Listing 7.9 : strreplace3.php

```
<?php
$phrases = array ("Jessicasse-croutes dans mon panier et
                   Jean ai d'autres dans mon placard",
                  "Jessicanapes dans mon salon et
                   Jean ai un dans la chambre");
echo $phrases[0]."<br />";
echo $phrases[1]."<br />";
$phrases = str_replace(array("Jessica","Jean"),
                       array("J'ai 6 ca", "j'en"), $phrases);
echo $phrases[0]."<br />";
echo $phrases[1]."<br />";
?>
```

Et le résultat :

```
Jessicasse-croutes dans mon panier et Jean ai d'autres dans mon placard
Jessicanapes dans mon salon et Jean ai un dans la chambre
J'ai 6 casse-croutes dans mon panier et j'en ai d'autres dans mon placard
J'ai 6 canapes dans mon salon et j'en ai un dans la chambre
```

Pour ne remplacer que des caractères, comme par exemple pour retirer les accents, on pourrait utiliser `str_replace()`, mais on peut également utiliser la fonction `strtr()`.


REMARQUE

Remplacement insensible à la casse

Si vous possedez la version 5.0.0 ou supérieure de PHP, vous pouvez utiliser la fonction `str_ireplace()` qui se comporte comme str_replace hormis qu'elle est remplacera également les chaînes de caractères ayant une casse différente. (minuscules/majuscules)

strtr() (syntaxe 1)

Retourne une chaîne de caractères dans laquelle certains caractères ont été remplacés par d'autres.

Syntaxe	`string strtr(string $chaine, string $depuis, string $vers)`
`$chaine`	Chaîne de caractères où les modifications doivent être effectuées.
`$depuis`	Chaîne de caractères indiquant tous les caractères à modifier.
`$vers`	Chaîne de caractères indiquant les caractères qui devront remplacer ceux de `$depuis`.
`retour`	La chaîne modifiée.

Voici un exemple qui permet de supprimer les accents dans une phrase.

Listing 7.10 : strtr1.php

```
<?php
$phrase = "Les accents sur à, é, è, ù, ê, ô vont être supprimés";
echo $phrase."<br />";
$phrase = strtr($phrase, "áéíóúàèìòùâêîôû", "aeiouaeiouaeiou");
echo $phrase;
?>
```

Voici le résultat obtenu :

```
Les accents sur à, é, è, ù, ê, ô vont être supprimés
Les accents sur a, e, e, u, e, o vont etre supprimes
```

Une autre syntaxe existe, qui consiste à déclarer un tableau associatif. Cela permet de remplacer des chaînes de caractères par d'autres.

strtr() (syntaxe 2)

Retourne une chaîne de caractères dans laquelle certains caractères ont été remplacés par d'autres.

Syntaxe	`string strtr(string $chaine, array $remplacement)`
`$chaine`	Chaîne de caractères où les modifications doivent être effectuées.
`$remplacement`	Tableau associatif ayant pour clés les chaînes de caractères à remplacer, et pour valeurs les chaînes de substitution correspondantes.
`retour`	La chaîne modifiée.

Voici un exemple utilisant cette syntaxe :

Listing 7.11 : strtr2.php

```
<?php
$phrase = "Les accents sur à, é, è, ù, ê, ô vont être supprimés";
echo $phrase."<br />";
$remplacements = array('à' => 'a',
                       'é' => 'e',
                       'è' => 'e',
                       'ù' => 'u',
                       'ê' => 'e',
                       'ô' => 'o',
                       'vont être' => 'ont étés' );
$phrase=strtr($phrase , $remplacements);
echo $phrase;
?>
```

Et voici le résultat obtenu :

```
Les accents sur à, é, è, ù, ê, ô vont être supprimés
Les accents sur a, e, e, u, e, o ont été supprimés
```

Fonctions statistiques (longueur et nombre d'occurrences)

strlen()

Retourne la longueur d'une chaîne de caractères.

Syntaxe	`int strlen(string $chaine)`
`$chaine`	Chaîne de caractères dont vous souhaitez connaître le nombre de caractères.
`retour`	Nombre de caractères de la chaîne.

```
<?php
    echo strlen("Texte de 22 caracteres");
?>
```

Et le résultat est :

```
22
```

substr_count()

Cette fonction compte le nombre d'occurrences d'une sous-chaîne dans une autre.

Syntaxe	`int substr_count(string $chaine, string $motif)`
`$chaine`	Chaîne de caractères où retrouver le motif.
`$motif`	Motif à rechercher.
`retour`	Le nombre de fois où le motif apparaît.

Listing 7.12 : c06-substrcount.php

```
<?php
echo substr_count("toto","toto");
echo "<br />";
echo substr_count("toto","to");
echo "<br />";
echo substr_count("Je me demande combien il y a de e
                  dans cette phrase","e");
echo "<br />";
?>
```

Et voici le résultat obtenu :

```
1
2
10
```

count_chars()

Permet de compter les occurrences des caractères dans une chaîne de caractères.

Syntaxe	`mixed count_chars(string $chaine [, int $mode])`
`$chaine`	Chaîne dont vous souhaitez compter le nombre d'occurrences des caractères.
`$mode`	Au choix : `0` (par défaut) renvoie le nombre d'occurrences de tous les caractères ayant un code ASCII compris entre 0 et 255. `1` ne renvoie que les caractères présents dans la chaîne. `2` ne renvoie que les caractères absents de la chaîne. `3` renvoie une chaîne de caractères des caractères utilisés. `4` renvoie une chaîne de caractères des caractères inutilisés.
`retour`	Un tableau associatif où les clés sont les codes ASCII des caractères, ou une chaîne de caractères selon le mode choisi.

Voici un exemple de script pour commenter cette fonction :

```
<?php
    $phrase="Cette phrase contient plusieurs e, t, c et p";
    echo "*** Code=0:\n";
    print_r(count_chars($phrase));
```

```
    echo "*** Code=1:\n";
    print_r(count_chars($phrase,1));
    echo "*** Code=2:\n";
    print_r(count_chars($phrase,2));
    echo "*** Code=3:\n";
    echo count_chars($phrase,3)."\n";
?>
```

Dont le résultat attendu est :

```
*** Code=0:
Array
(
    [0] => 0    [1] => 0    [2] => 0    [3] => 0    [4] => 0
    [5] => 0    [6] => 0    [7] => 0    [8] => 0    [9] => 0
    [10] => 0    [11] => 0    [12] => 0    [13] => 0    [14] => 0
    [15] => 0    [16] => 0    [17] => 0    [18] => 0    [19] => 0
    [20] => 0    [21] => 0    [22] => 0    [23] => 0    [24] => 0
    [25] => 0    [26] => 0    [27] => 0    [28] => 0    [29] => 0
    [30] => 0    [31] => 0    [32] => 8    [33] => 0    [34] => 0
    [35] => 0    [36] => 0    [37] => 0    [38] => 0    [39] => 0
    [40] => 0    [41] => 0    [42] => 0    [43] => 0    [44] => 2
    [45] => 0    [46] => 0    [47] => 0    [48] => 0    [49] => 0
    [50] => 0    [51] => 0    [52] => 0    [53] => 0    [54] => 0
    [55] => 0    [56] => 0    [57] => 0    [58] => 0    [59] => 0
    [60] => 0    [61] => 0    [62] => 0    [63] => 0    [64] => 0
    [65] => 0    [66] => 0    [67] => 1    [68] => 0    [69] => 0
    [70] => 0    [71] => 0    [72] => 0    [73] => 0    [74] => 0
    [75] => 0    [76] => 0    [77] => 0    [78] => 0    [79] => 0
    [80] => 0    [81] => 0    [82] => 0    [83] => 0    [84] => 0
    [85] => 0    [86] => 0    [87] => 0    [88] => 0    [89] => 0
    [90] => 0    [91] => 0    [92] => 0    [93] => 0    [94] => 0
    [95] => 0    [96] => 0    [97] => 1    [98] => 0    [99] => 2
    [100] => 0    [101] => 7    [102] => 0    [103] => 0    [104] => 1
    [105] => 2    [106] => 0    [107] => 0    [108] => 1    [109] => 0
    [110] => 2    [111] => 1    [112] => 3    [113] => 0    [114] => 2
    [115] => 3    [116] => 6    [117] => 2    [118] => 0    [119] => 0
    [120] => 0    [121] => 0    [122] => 0    [123] => 0    [124] => 0
    [125] => 0    [126] => 0    [127] => 0    [128] => 0    [129] => 0
    [130] => 0    [131] => 0    [132] => 0    [133] => 0    [134] => 0
    [135] => 0    [136] => 0    [137] => 0    [138] => 0    [139] => 0
    [140] => 0    [141] => 0    [142] => 0    [143] => 0    [144] => 0
    [145] => 0    [146] => 0    [147] => 0    [148] => 0    [149] => 0
    [150] => 0    [151] => 0    [152] => 0    [153] => 0    [154] => 0
    [155] => 0    [156] => 0    [157] => 0    [158] => 0    [159] => 0
    [160] => 0    [161] => 0    [162] => 0    [163] => 0    [164] => 0
    [165] => 0    [166] => 0    [167] => 0    [168] => 0    [169] => 0
    [170] => 0    [171] => 0    [172] => 0    [173] => 0    [174] => 0
    [175] => 0    [176] => 0    [177] => 0    [178] => 0    [179] => 0
    [180] => 0    [181] => 0    [182] => 0    [183] => 0    [184] => 0
    [185] => 0    [186] => 0    [187] => 0    [188] => 0    [189] => 0
    [190] => 0    [191] => 0    [192] => 0    [193] => 0    [194] => 0
```

```
    [195] => 0    [196] => 0    [197] => 0    [198] => 0    [199] => 0
    [200] => 0    [201] => 0    [202] => 0    [203] => 0    [204] => 0
    [205] => 0    [206] => 0    [207] => 0    [208] => 0    [209] => 0
    [210] => 0    [211] => 0    [212] => 0    [213] => 0    [214] => 0
    [215] => 0    [216] => 0    [217] => 0    [218] => 0    [219] => 0
    [220] => 0    [221] => 0    [222] => 0    [223] => 0    [224] => 0
    [225] => 0    [226] => 0    [227] => 0    [228] => 0    [229] => 0
    [230] => 0    [231] => 0    [232] => 0    [233] => 0    [234] => 0
    [235] => 0    [236] => 0    [237] => 0    [238] => 0    [239] => 0
    [240] => 0    [241] => 0    [242] => 0    [243] => 0    [244] => 0
    [245] => 0    [246] => 0    [247] => 0    [248] => 0    [249] => 0
    [250] => 0    [251] => 0    [252] => 0    [253] => 0    [254] => 0
    [255] => 0
)
*** Code=1:
Array
(
    [32] => 8    [44] => 2    [67] => 1    [97] => 1    [99] => 2
    [101] => 7    [104] => 1    [105] => 2    [108] => 1    [110] => 2
    [111] => 1    [112] => 3    [114] => 2    [115] => 3    [116] => 6
    [117] => 2
)
*** Code=2:
Array
(
    [0] => 0    [1] => 0    [2] => 0    [3] => 0    [4] => 0
    [5] => 0    [6] => 0    [7] => 0    [8] => 0    [9] => 0
    [10] => 0    [11] => 0    [12] => 0    [13] => 0    [14] => 0
    [15] => 0    [16] => 0    [17] => 0    [18] => 0    [19] => 0
    [20] => 0    [21] => 0    [22] => 0    [23] => 0    [24] => 0
    [25] => 0    [26] => 0    [27] => 0    [28] => 0    [29] => 0
    [30] => 0    [31] => 0    [33] => 0    [34] => 0    [35] => 0
    [36] => 0    [37] => 0    [38] => 0    [39] => 0    [40] => 0
    [41] => 0    [42] => 0    [43] => 0    [45] => 0    [46] => 0
    [47] => 0    [48] => 0    [49] => 0    [50] => 0    [51] => 0
    [52] => 0    [53] => 0    [54] => 0    [55] => 0    [56] => 0
    [57] => 0    [58] => 0    [59] => 0    [60] => 0    [61] => 0
    [62] => 0    [63] => 0    [64] => 0    [65] => 0    [66] => 0
    [68] => 0    [69] => 0    [70] => 0    [71] => 0    [72] => 0
    [73] => 0    [74] => 0    [75] => 0    [76] => 0    [77] => 0
    [78] => 0    [79] => 0    [80] => 0    [81] => 0    [82] => 0
    [83] => 0    [84] => 0    [85] => 0    [86] => 0    [87] => 0
    [88] => 0    [89] => 0    [90] => 0    [91] => 0    [92] => 0
    [93] => 0    [94] => 0    [95] => 0    [96] => 0    [98] => 0
    [100] => 0    [102] => 0    [103] => 0    [106] => 0    [107] => 0
    [109] => 0    [113] => 0    [118] => 0    [119] => 0    [120] => 0
    [121] => 0    [122] => 0    [123] => 0    [124] => 0    [125] => 0
    [126] => 0    [127] => 0    [128] => 0    [129] => 0    [130] => 0
    [131] => 0    [132] => 0    [133] => 0    [134] => 0    [135] => 0
    [136] => 0    [137] => 0    [138] => 0    [139] => 0    [140] => 0
    [141] => 0    [142] => 0    [143] => 0    [144] => 0    [145] => 0
    [146] => 0    [147] => 0    [148] => 0    [149] => 0    [150] => 0
```

```
    [151] => 0    [152] => 0    [153] => 0    [154] => 0    [155] => 0
    [156] => 0    [157] => 0    [158] => 0    [159] => 0    [160] => 0
    [161] => 0    [162] => 0    [163] => 0    [164] => 0    [165] => 0
    [166] => 0    [167] => 0    [168] => 0    [169] => 0    [170] => 0
    [171] => 0    [172] => 0    [173] => 0    [174] => 0    [175] => 0
    [176] => 0    [177] => 0    [178] => 0    [179] => 0    [180] => 0
    [181] => 0    [182] => 0    [183] => 0    [184] => 0    [185] => 0
    [186] => 0    [187] => 0    [188] => 0    [189] => 0    [190] => 0
    [191] => 0    [192] => 0    [193] => 0    [194] => 0    [195] => 0
    [196] => 0    [197] => 0    [198] => 0    [199] => 0    [200] => 0
    [201] => 0    [202] => 0    [203] => 0    [204] => 0    [205] => 0
    [206] => 0    [207] => 0    [208] => 0    [209] => 0    [210] => 0
    [211] => 0    [212] => 0    [213] => 0    [214] => 0    [215] => 0
    [216] => 0    [217] => 0    [218] => 0    [219] => 0    [220] => 0
    [221] => 0    [222] => 0    [223] => 0    [224] => 0    [225] => 0
    [226] => 0    [227] => 0    [228] => 0    [229] => 0    [230] => 0
    [231] => 0    [232] => 0    [233] => 0    [234] => 0    [235] => 0
    [236] => 0    [237] => 0    [238] => 0    [239] => 0    [240] => 0
    [241] => 0    [242] => 0    [243] => 0    [244] => 0    [245] => 0
    [246] => 0    [247] => 0    [248] => 0    [249] => 0    [250] => 0
    [251] => 0    [252] => 0    [253] => 0    [254] => 0    [255] => 0
)
*** Code=3:
 ,Cacehilnoprstu
```

strspn()

Cette fonction retourne le nombre de caractères du début de la chaîne passée en premier paramètre, dont tous les caractères sont compris dans la chaîne fournie en second paramètre.

Syntaxe	`int strspn(string $chaine, string $caracteres [, int $debut [, int $nbCaracteres]])`
`$chaine`	Chaîne à étudier.
`$caracteres`	Ensemble de caractères.
`$debut`	Indice du premier caractère de `$chaine` a étudier.
`$nbCaracteres`	Nombre de caractères a partir de `$debut` sur lesquels faire le compte.
`retour`	Le nombre de caractères du début de la chaîne, dont tous les caractères appartiennent à `$caracteres`.

Un exemple :

```
<?php
    echo strspn("7490875253 Voila 10 chiffres", "1234567890");
?>
```

dont le résultat est :

```
10
```

strcspn()

Cette fonction retourne le nombre de caractères du début de la chaîne passée en premier paramètre, dont aucun caractère n'est compris dans la chaîne fournie en second paramètre.

Syntaxe	`int strcspn(string $chaine, string $caracteres [ int $debut [, int $nbCaracteres]])`
`$chaine`	Chaîne à étudier.
`$caracteres`	Ensemble de caractères.
`$debut`	Indice du premier caractère de `$chaine` a étudier.
`$nbCaracteres`	Nombre de caractères a partir de `$debut` sur lesquels faire le compte.
`retour`	Le nombre de caractères du début de la chaîne, dont aucun caractère n'appartient à `$caracteres`.

Un exemple :

```
<?php
    echo strcspn("22 caracteres avant le: ou ! ou ?", ":!?");
?>
```

avec comme résultat :

```
22
```

Fonctions de position

Il est également utile de pouvoir récupérer l'index d'un caractère dans une chaîne :

strpos()

Retourne la position du premier caractère de la première occurrence d'une chaîne dans une autre, à partir d'un certain index.

Syntaxe	`int strpos(string $chaine, string $souschaine [int $index])`
`$chaine`	Chaîne de caractères dans laquelle rechercher la sous-chaîne.
`$souschaine`	Chaîne de caractères à rechercher.
`$index`	Index de la chaîne à partir duquel la recherche doit s'effectuer.
`retour`	L'index du premier caractère de la sous-chaîne. Retourne `FALSE` si la sous-chaîne n'est pas trouvée.

```
<?php
echo strpos("Index du premier caractère d'une chaîne de".
            " caractères","caractère");
```

```
echo "\n";
echo strpos("Index du premier caractère d'une chaîne de".
            " caractères","caractère",0);
echo "\n";
echo strpos("Index du premier caractère d'une chaine de".
            " caractères","caractère",30);
?>
```

Voici le résultat obtenu :

```
17
17
43
```

Retour de la fonction
Il vaut mieux comparer le résultat à `FALSE` *(en utilisant l'opérateur* `===`*) avant de se servir de la valeur retournée, car la confusion entre le premier caractère (d'indice 0) et la valeur* `FALSE` *est possible.*

strrpos()

Retourne la position de la dernière occurrence d'un caractère dans une chaîne de caractères.

Syntaxe	`int strrpos(string $chaine, char $caractere)`
`$chaine`	Chaîne dans laquelle rechercher le caractère.
`$caractere`	Caractère dont on veut l'indice.
`retour`	La position de la dernière occurrence du caractère dans la chaîne.

```
<?php
echo strrpos("Index du premier caractere d'une chaine de caracteres",'e');
?>
```

Voici le résultat obtenu :

```
51
```


7.3. Comparaison de chaînes de caractères

Il est possible de comparer deux chaînes de caractères en utilisant simplement l'opérateur `==`. Cependant, il peut être également intéressant de comparer des chaînes semblables à l'aide de fonctions permettant d'autres moyens de comparaison que la stricte égalité de deux chaînes.


ATTENTION

Égalite des valeurs et des types
Si vous souhaitez comparer une variable à une chaîne de caractères en utilisant l'opérateur ==, assurez-vous que la variable est bien de type `string`*. Vous risqueriez, sinon, d'avoir des surprises.*

Comparaison par ordre alphabétique

Il est ainsi possible de comparer deux chaînes selon leur ordre alphabétique.

strcmp()

Compare deux chaînes de caractères à partir de l'ordre alphabétique.

Syntaxe	`int strcmp(string $chaine1, string $chaine2)`
`$chaine1`	Chaîne à comparer.
`$chaine2`	Chaîne à comparer.
`retour`	`-1` si la chaîne 1 est placée avant la chaîne 2 dans l'ordre alphabétique, `1` dans le cas contraire et `0` si les deux chaînes sont identiques.

```
<?php
echo strcmp("abc", "bcd")."\n";
echo strcmp("aa", "aaa")."\n";
echo strcmp("_mot", "mot")."\n";
echo strcmp("mot_", "mot")."\n";
echo strcmp("mot", "mot")."\n";
?>
```

Voici le résultat obtenu :

```
-1
-1
-1
1
0
```

Pour ne comparer que les premiers caractères, il suffit d'utiliser la fonction suivante :

strncmp()

Compare les premiers caractères de deux chaînes de caractères à partir de l'ordre alphabétique.

Syntaxe	`int strncmp(string $chaine1, string $chaine2, int $nbCaracteres)`
`$chaine1`	Chaîne à comparer.
`$chaine2`	Chaîne à comparer.
`$nbCaracteres`	Nombre de caractères à comparer.
`retour`	`-1` si les `$nbCaracteres` premiers caractère de chaîne 1 sont placés avant la chaîne 2 dans l'ordre alphabétique, `1` dans le cas contraire et `0` si les `$nbCaracteres` premiers caractères des deux chaînes sont identiques.

```
<?php
echo strncmp("aa", "aaa", 2)."\n";
echo strncmp("aa", "aaa", 3)."\n";
echo strncmp("abcdg", "abcef", 3)."\n";
echo strncmp("abcdg", "abcef", 4)."\n";
?>
```

Le résultat obtenu est alors le suivant :

```
0
-1
0
-1
```

strcoll()

Compare deux chaînes de caractères selon la configuration locale du serveur. Si la configuration courante est C ou POSIX, alors cette fonction est identique à `strcmp()`.

Syntaxe `int strcoll(string $chaine1, string $chaine2)`

Les fonctions `strcmp()` et `strncmp()` ont leurs équivalents insensibles à la casse ; ce sont les fonctions `strcasecmp()` et `strncasecmp()`.

strcasecmp()

Compare deux chaînes de caractères à partir de l'ordre alphabétique sans tenir compte de la casse.

Syntaxe	`int strcasecmp(string $chaine1, string $chaine2)`
`$chaine1`	Chaîne à comparer.
`$chaine2`	Chaîne à comparer.
`retour`	`-1` si la chaîne 1 est placée avant la chaîne 2 dans l'ordre alphabétique, `1` dans le cas contraire et `0` si les deux chaînes sont identiques.

strncasecmp()

Compare les premiers caractères de deux chaînes de caractères à partir de l'ordre alphabétique sans tenir compte de la casse.

Syntaxe	`int strncasecmp(string $chaine1, string $chaine2, int $nbCaracteres)`
`$chaine1`	Chaîne à comparer.
`$chaine2`	Chaîne à comparer.
`$nbCaracteres`	Nombre de caractères à comparer.
`retour`	`-1` si les `$nbCaracteres` premiers caractère de chaîne 1 sont placés avant la chaîne 2 dans l'ordre alphabétique, `1` dans le cas contraire et `0` si les `$nbCaracteres` premiers caractères des deux chaînes sont identiques.

Un algorithme de "comparaison naturelle" est également disponible. Voici sa description :

strnatcmp()

Compare deux chaînes utilisant un algorithme censé ordonner des chaînes de caractères comme le ferait un être humain.

Syntaxe	`int strnatcmp(string $chaine1, string $chaine2)`
`$chaine1`	Chaîne à comparer.
`$chaine2`	Chaîne à comparer.
`retour`	`-1` si la chaîne 1 est placée avant la chaîne 2 dans l'ordre "naturel", `1` dans le cas contraire et `0` si les deux chaînes sont identiques.

Pour mettre en avant la différence entre `strcmp()` et `strnatcmp()`, le script suivant trie un tableau selon les deux méthodes.

```
<?php
$tableau1 = $tableau2  = array ('image5.jpg',
                                'image4.jpg',
                                'image12.jpg',
                                'image8.jpg',
                                'image1.jpg',
                                'image43.jpg',
                                'image14.jpg');
echo "Ordre standard:\n";
usort($tableau1, "strcmp");
print_r($tableau1);
echo "\nOrdre naturel:\n";
usort($tableau2, "strnatcmp");
print_r($tableau2);
?>
```

Voici le résultat obtenu :

```
Ordre standard:
Array
(
    [0] => image1.jpg
    [1] => image12.jpg
    [2] => image14.jpg
    [3] => image4.jpg
    [4] => image43.jpg
    [5] => image5.jpg
    [6] => image8.jpg
)

Ordre naturel:
Array
(
    [0] => image1.jpg
    [1] => image4.jpg
    [2] => image5.jpg
    [3] => image8.jpg
    [4] => image12.jpg
    [5] => image14.jpg
    [6] => image43.jpg
)
```

Une fonction équivalente, mais insensible à la casse, existe. Elle s'appelle `strnatcasecmp()`.

strnatcasecmp()

Compare deux chaînes utilisant un algorithme censé ordonner des chaînes de caractères comme le ferait un être humain. Cette fonction est insensible à la casse.

Syntaxe	`int strnatcasecmp(string $chaine1, string $chaine2)`
`$chaine1`	Chaîne à comparer.
`$chaine2`	Chaîne à comparer.
`retour`	`-1` si la chaîne 1 est placée avant la chaîne 2 dans l'ordre "naturel", `1` dans le cas contraire et `0` si les deux chaînes sont identiques.

```
<?php
$tableau1 = $tableau2  = array ('Image5.jpg',
                                'image4.jpg',
                                'Image12.jpg',
                                'image8.jpg',
                                'Image1.jpg',
                                'image43.jpg',
                                'Image14.jpg');
echo "Ordre standard:\n";
```

```
usort($tableau1, "strcmp");
print_r($tableau1);
echo "\nOrdre naturel:\n";
usort($tableau2, "strnatcasecmp");
print_r($tableau2);
?>
```

Voici le résultat obtenu :

```
Ordre standard:
Array
(
    [0] => Image1.jpg
    [1] => Image12.jpg
    [2] => Image14.jpg
    [3] => image4.jpg
    [4] => image43.jpg
    [5] => Image5.jpg
    [6] => image8.jpg
)

Ordre naturel:
Array
(
    [0] => Image1.jpg
    [1] => image4.jpg
    [2] => Image5.jpg
    [3] => image8.jpg
    [4] => Image12.jpg
    [5] => Image14.jpg
    [6] => image43.jpg
)
```

Vous pouvez également utiliser les fonctions `min()` *et* `max()`*, décrites dans le chapitre "PHP et les mathématiques", pour déterminer la chaîne de caractères ayant la première ou la dernière position dans un classement alphabétique d'une liste de chaînes.*

Comparaison orthographique

similar_text()

Permet de comparer deux chaînes en estimant leurs ressemblances. Cette fonction est sensible à la casse.

Syntaxe	`int similar_text(string $chaine1, string $chaine2, [double &$pourcentage])`
`$chaine1`	Une des deux chaînes à comparer.
`$chaine2`	La chaîne avec laquelle on veut comparer la première.
`$pourcentage`	Si une référence est passée en paramètre, la valeur en % y sera déclarée.
`retour`	Le nombre de caractères en commun.

```
<?php
$chaine1 = "Une des chaines a comparer";
$chaine2 = "L'autre des chaines a comparer";
echo similar_text($chaine1, $chaine2)."\n";
$pourcentage = 0;
echo similar_text($chaine1, $chaine2, &$pourcentage)."\n";
echo $pourcentage."\n";
echo similar_text("aaa", "AAA", &$pourcentage)."\n";
echo $pourcentage."\n";
?>
```

Voici le résultat obtenu :

```
24
24
85.714285714286
0
0
```

Méthode utilisée
La méthode utilisée est celle de Oliver [1993] qui est décrite à l'adresse suivante : ***http://citeseer.nj.nec.com/oliver93decision.html****.*

Il existe une autre méthode permettant de mesurer la distance entre deux chaînes de caractères, c'est la distance de Levenshtein. Le calcul est ici moins gourmand que le précédent.

levenshtein()

Calcule la distance de Levenshtein entre deux chaînes de caractères. La distance de Levenshtein se définit comme étant le plus petit nombre de caractères à remplacer dans la première chaîne pour obtenir la seconde.

Syntaxe	`int levenshtein(string $chaine1, string $chaine2 [, int $coutInsert, int $coutRemplace, int $coutSupprime])`
`$chaine1`	Chaîne à comparer.
`$chaine2`	Chaîne à comparer.
`$coutInsert`	Coût d'une insertion.

`$coutRemplace`	Coût d'un remplacement.
`$coutSupprime`	Coût d'une suppression.
`retour`	Distance de Levenshtein, ou `1` si l'une des deux chaînes fait plus de 255 caractères.

```
<?php
$chaine1 = "Une des chaînes à comparer";
$chaine2 = "L'autre des chaînes à comparer";
echo levenshtein($chaine1, $chaine2)."\n";
echo levenshtein($chaine1, $chaine2, 2, 3, 1)."\n";
echo levenshtein("aaa", "AAA")."\n";
?>
```

Voici le résultat obtenu :

```
5
11
3
```

Comparaison phonique

Les fonctions présentées ici ne permettent pas une comparaison directe, mais permettent de définir une valeur basée sur la consonance d'un mot.

soundex()

Permet de calculer une valeur à partir de la prononciation d'une chaîne de caractères. La particularité de cette valeur est que deux mots ayant la même consonance auront le même "soundex". Attention, cette fonction est adaptée à la prononciation anglaise.

Syntaxe	`string soundex(string $chaine)`
`$chaine`	Chaîne de caractères dont on veut calculer la valeur `'soundex'`.
`retour`	Le "soundex".

Listing 7.13 : soundex.php

```
<?php
    echo soundex("serial killer");
    echo "<br />";
    echo soundex("seriol quilleur");
    echo "<br />";
    echo soundex("Welcome");
    echo "<br />";
    echo soundex("ouelcome");
    echo "<br />";
    echo soundex("elephant");
```

```
    echo "<br />";
    echo soundex("elefant");
    echo "<br />";
    echo soundex("elefante");
?>
```

Le résultat de ce script est le suivant :

```
S642
S642
W425
O425
E415
E415
E415
```

metaphone()

Cette fonction est très similaire à `soundex()`. Elle a le même objectif, mais utilise une représentation et un algorithme différents.

Syntaxe	`string metaphone(string $chaine)`
`$chaine`	Chaîne de caractères dont vous souhaitez le "metaphone".
`retour`	Une valeur dépendant de la façon de prononcer la chaîne de caractères.

Listing 7.14 : metaphone.php

```
<?php
    echo metaphone("serial killer");
    echo "<br />";
    echo metaphone("seriol quilleur");
    echo "<br />";
    echo metaphone("Welcome");
    echo "<br />";
    echo metaphone("ouelcome");
    echo "<br />";
    echo metaphone("elephant");
    echo "<br />";
    echo metaphone("elefant");
    echo "<br />";
    echo metaphone("elefante");
?>
```

Dont le résultat est :

```
SRLKLR
SRLKLR
WLKM
```

```
OLKM
ELFNT
ELFNT
ELFNT
```

7.4. Gestion des caractères spéciaux

Ajout du caractère d'échappement

Il est parfois utile d'encoder des chaînes de caractères. C'est le cas, par exemple, pour échapper certains caractères spéciaux.

addSlashes()

Permet l'échappement de certains caractères. Concrètement, il s'agit des caractères : `'`, `"`, `\` et `NUL` (`\0`).

Syntaxe	`string addSlashes(string $chaine)`
`$chaine`	Chaîne de caractères à modifier.
`retour`	La chaîne avec les caractères spéciaux échappés.

```
<?php
  $chaine="C'est \ Cool";
  echo $chaine;
  echo "\n";
  echo addSlashes($chaine);
?>
```

Voici le résultat obtenu :

```
C'est \ Cool
C\'est \\ Cool
```


REMARQUE

magic_quotes_gpc
Par défaut, les 'magic quotes' (apostrophes magiques) sont activées, c'est-à-dire que les apostrophes sont automatiquement précédées du signe `\` lorsque les valeurs sont passées à un script par la méthode `GET`, `POST` ou par cookie.

Une fonction très similaire, appelée `quoteMeta()`, permet également l'échappement de certains caractères.

quoteMeta()

Permet d'échapper les caractères : `.`, `\`, `+`, `?`, `^`, `$`, `[`, `]`, `(`, `)`.

Syntaxe	`string quoteMeta(string $chaine)`
`$chaine`	Chaîne à transformer.
`retour`	Chaîne transformée.

addCSlashes()

Retourne une chaîne de caractères en ajoutant des `\` devant les caractères précisés. Les caractères ayant un code ASCII inférieur à 32, ou supérieur à 126, sont convertis à leur valeur octale.

Syntaxe	`string addCSlashes(string $chaine, string $listeCaracteres)`
`$chaine`	Chaîne de caractères à transformer.
`$listeCaracteres`	Liste des caractères à échapper.
`retour`	Chaîne transformée.

```
<?php
  $chaine="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
  echo $chaine;
  echo "\n";
  echo addcslashes($chaine,"A..z");
  echo "\n";
  echo addcslashes($chaine,"G..f");
?>
```

Et voici le résultat obtenu :

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
\A\B\C\D\E\F\G\H\I\J\K\L\M\N\O\P\Q\R\S\T\U\V\W\X\Y\Z\a\b\c\d\e\f\g\h\i\j\k\l\m\n
\o\p\q\r\s\t\u\v\w\x\y\z
ABCDEF\G\H\I\J\K\L\M\N\O\P\Q\R\S\T\U\V\W\X\Y\Z\a\b\c\d\e\fghijklmnopqrstuvwxyz
```

Caractères spéciaux
Le fait d'ajouter un `\` peut transformer certaines caractères en caractères spéciaux, c'est le cas de `0`, `a`, `b`, `f`, `n`, `r`, `t` et `v`.

Suppression du caractère d'échappement

Les fonctions inverses existent bien évidemment. Elles permettent de supprimer les `\`.

stripSlashes()

Retire les slashes ajoutés par la fonction `addSlashes()`.

Syntaxe	`string stripSlashes(string $chaine)`
`$chaine`	Chaîne de caractères pour laquelle vous souhaitez retirer les `\`.
`retour`	La chaîne sans les `\` d'échappement.

```
<?php
  $chaine="Cette chaine contient des ' et des \.";
  echo $chaine;
  echo "\n";
  echo addSlashes($chaine);
  echo "\n";
  echo stripSlashes(addslashes($chaine));
?>
```

Ce qui donne en retour :

```
Cette chaine contient des ' et des \.
Cette chaine contient des \' et des \\.
Cette chaine contient des ' et des \.
```

De la même manière, le résultat obtenu par `addCSlashes()` est obtenu par `stripCSlashes()`.

stripCSlashes()

Fonction inverse de `addCSlashes()`.

Syntaxe	`string stripCSlashes(string $chaine)`
`$chaine`	Chaîne dont vous souhaitez retirer les `\` d'échappement.
`retour`	Chaîne sans les `\` d'échappement.

Dans l'exemple suivant, nous ne nous sommes pas méfiés des caractères spéciaux. Observez le résultat obtenu :

```
<?php
  $chaine="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
  echo $chaine;
  echo "\n";
  echo addCSlashes($chaine,"G..f");
  echo "\n";
  echo stripCSlashes(addCSlashes($chaine,"G..f"));
?>
```

Résultat obtenu :

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
ABCDEF\G\H\I\J\K\L\M\N\O\P\Q\R\S\T\U\V\W\X\Y\Z\a\b\c\d\e\fghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ□cde□ghijklmnopqrstuvwxyz
```

Conversion des caractères en code HTML

Dans le cas de l'écriture de pages HTML, il peut être utile de transformer certains caractères spéciaux en leurs équivalents HTML. C'est le cas des caractères `&`, `"`, `'`, < et >, qui peuvent être remplacés respectivement par `&`, `&quot;`, `&apos;`, `<` et `>`.

htmlSpecialChars()

Retourne une chaîne de caractères pour laquelle les caractères spéciaux de l'HTML ont été convertis. Cela ne concerne pas les caractères accentués.

Syntaxe	`string htmlSpecialChars(string $chaine [, int $apostrophes [, string $encodage]])`
`$chaine`	Chaîne de caractères à transformer.
`$apostrophes`	Au choix : ENT_COMPAT (par défaut), pour que cette fonction ne transforme que les guillemets et laisse les apostrophes telles quelles. `ENT_QUOTES`, pour transformer les apostrophes et guillemets. `ENT_NOQUOTES`, pour ne transformer ni les apostrophes ni les guillemets.
`$encodage`	Par défaut il vaut "ISO-8859-1".
`retour`	La chaîne transformée.


REMARQUE

Utilité
Cette fonction est notamment utile pour les forums ou livres d'or, par exemple. En effet, si un utilisateur utilise l'un de ces caractères, il est souhaitable que ceux-ci réapparaissent tels quels par la suite.

Voici un exemple :

Listing 7.15 : htmlspecialchars.php

```
<html>
<head><title>HTMLSpecialChars</title></head>
<body>
<p>
<?php
  $chaine = "Un message avec du HTML <i> &, è, \"  et des ' </i>";
```

```
  echo $chaine."<br />\n";
  echo htmlspecialchars($chaine);
?>
</p>
</body>
</html>
```

Voici le code source obtenu :

```
<html>
<head><title>HTMLSpecialChars</title></head>
<body>
<p>
Un message avec du HTML <i> &, è, "  et des ' </i><br />
Un message avec du HTML &lt;i&gt; &amp;, è, &quot;  et des ' &lt;/i&gt;</p>
</body>
</html>
```

Et, donc, le résultat du navigateur :

```
Un message avec du HTML &, è, " et des '
Un message avec du HTML <i> &, è, " et des ' </i>
```

Une autre fonction très similaire permet de transformer tous les caractères spéciaux en leurs équivalents HTML. Elle s'appelle `htmlEntities()`.

htmlEntities()

Retourne une chaîne de caractères pour laquelle tous les caractères spéciaux (y compris les caractères accentués) ont été convertis en leurs équivalents HTML.

Syntaxe	`string htmlEntities(string $chaine[, int $apostrophes [, string $encodage]])`
`$chaine`	Chaîne de caractères à transformer.
`$apostrophes`	Au choix : `ENT_COMPAT` (par défaut), pour que cette fonction ne transforme que les guillemets et laisse les apostrophes telles quelles. `ENT_QUOTES`, pour transformer les apostrophes et les guillemets. `ENT_NOQUOTES`, pour ne transformer ni les apostrophes ni les guillemets.
`$encodage`	Par défaut, il vaut "ISO-8859-1".
`retour`	La chaîne transformée.

Voici le script d'exemple :

Listing 7.16 : htmlentities.php

```
<html>
<head><title>HTMLEntities</title></head>
<body>
<p>
<?php
  $chaine = "Un message avec du HTML <i> &, è, \"  et des ' </i>";
  echo $chaine."<br />\n";
  echo htmlEntities($chaine);
?>
</p>
</body>
</html>
```

Voici le résultat obtenu à l'écran :

```
Un message avec du HTML &, è, " et des '
Un message avec du HTML <i> &, è, " et des ' </i>
```

et voici le code source correspondant :

```
<html>
<head><title>HTMLEntities</title></head>
<body>
<p>
Un message avec du HTML <i> &, è, "  et des ' </i><br />
Un message avec du HTML &lt;i&gt; &amp;, &egrave;, &quot;  et des '
✂ &lt;/i&gt;</p>
</body>
</html>
```


get_html_translation_table()

Permet de placer, dans un tableau associatif, les tables de conversion des caractères spéciaux en leurs équivalents HTML, utilisées par les fonctions `htmlSpecialChars()` et `htmlEntities()`.

Syntaxe	`array get_html_translation_table(int $table [, int $apostrophes])`
`$table`	Table à récupérer, au choix : `HTML_ENTITIES`, pour la table utilisée par `htmlEntities()`. `HTML_SPECIALCHARS`, pour la table utilisée par `htmlSpecialChars()`.
`$apostrophes`	Au choix : `ENT_COMPAT` (par défaut), pour que cette fonction ne transforme que les guillemets et laisse les apostrophes telles quelles.

	ENT_QUOTES, pour transformer les apostrophes et les guillemets. ENT_NOQUOTES, pour ne transformer ni les apostrophes ni les guillemets.
retour	Un tableau associatif ayant pour clés les caractères spéciaux, et pour valeurs leurs équivalents HTML.

Voici un script d'exemple :

Listing 7.17 : get_html_translation_table.php

```
<?php
    print_r(get_html_translation_table(HTML_ENTITIES));
    print_r(get_html_translation_table(HTML_SPECIALCHARS));
?>
```

dont le résultat est :

```
Array
(
    [ ] =>  
    [¡] => &iexcl;
    [¢] => &cent;
    [£] => &pound;
    [¤] => &curren;
    [¥] => &yen;
    [¦] => &brvbar;
    [§] => &sect;
    [¨] => &uml;
    [©] => &copy;
    [ª] => &ordf;
    [« ] => &laquo;
    [¬] => &not;
    [-] => &shy;
    [®] => &reg;
    [¯] => &macr;
    [°] => &deg;
    [±] => &plusmn;
    [²] => &sup2;
    [³] => &sup3;
    [´] => &acute;
    [µ] => &micro;
    [¶] => &para;
    [·] => &middot;
    [¸] => &cedil;
    [¹] => &sup1;
    [º] => &ordm;
    [ »] => &raquo;
    [¼] => &frac14;
    [½] => &frac12;
    [¾] => &frac34;
    [¿] => &iquest;
    [À] => &Agrave;
```

```
[Á] => &Aacute;
[Â] => &Acirc;
[Ã] => &Atilde;
[Ä] => &Auml;
[Å] => &Aring;
[Æ] => &AElig;
[Ç] => &Ccedil;
[È] => &Egrave;
[É] => &Eacute;
[Ê] => &Ecirc;
[Ë] => &Euml;
[Ì] => &Igrave;
[Í] => &Iacute;
[Î] => &Icirc;
[Ï] => &Iuml;
[Ð] => &ETH;
[Ñ] => &Ntilde;
[Ò] => &Ograve;
[Ó] => &Oacute;
[Ô] => &Ocirc;
[Õ] => &Otilde;
[Ö] => &Ouml;
[×] => &times;
[Ø] => &Oslash;
[Ù] => &Ugrave;
[Ú] => &Uacute;
[Û] => &Ucirc;
[Ü] => &Uuml;
[Ý] => &Yacute;
[Þ] => &THORN;
[ß] => &szlig;
[à] => &agrave;
[á] => &aacute;
[â] => &acirc;
[ã] => &atilde;
[ä] => &auml;
[å] => &aring;
[æ] => &aelig;
[ç] => &ccedil;
[è] => &egrave;
[é] => &eacute;
[ê] => &ecirc;
[ë] => &euml;
[ì] => &igrave;
[í] => &iacute;
[î] => &icirc;
[ï] => &iuml;
[ð] => &eth;
[ñ] => &ntilde;
[ò] => &ograve;
[ó] => &oacute;
[ô] => &ocirc;
```

```
    [õ] => &otilde;
    [ö] => &ouml;
    [÷] => &divide;
    [ø] => &oslash;
    [ù] => &ugrave;
    [ú] => &uacute;
    [û] => &ucirc;
    [ü] => &uuml;
    [ý] => &yacute;
    [þ] => &thorn;
    [&] => &amp;
    ["] => &quot;
    [<] => &lt;
    [>] => &gt;
)
Array
(
    [&] => &amp;
    ["] => &quot;
    [<] => &lt;
    [>] => &gt;
)
```

nl2br()

Retourne une chaîne de caractères dans laquelle les retours chariot ont été transformés en balises de retours de lignes HTML (`<br />`).

Syntaxe	`string nl2br(string $chaine)`
`$chaine`	Chaîne de caractères à transformer.
`retour`	La chaîne transformée.

Listing 7.18 : nl2br.php

```
<html>
<head><title>nl2br</title></head>
<body>
<p>
<?php
  $chaine = "Un message avec
des retours
de ligne";
  echo $chaine."<br />\n";
  echo nl2br($chaine);
?>
</p>
</body>
</html>
```

Voici la sortie à l'écran :

```
Un message avec des retours de ligne
Un message avec
des retours
de ligne
```

Et voici le code HTML généré :

```
<html>
<head><title>nl2br</title></head>
<body>
<p>
Un message avec
des retours
de ligne<br />
Un message avec
<br />
des retours
<br />
de ligne</p>
</BODY>
</HTML>
```


REMARQUE

Modification depuis PHP 4.3.2
Le comportement de cette fonction a légèrement été modifié pour supporter tout types de retours à la ligne. D'anciens scripts peuvent donc se comporter différement sur une version récente de PHP.

Conversion d'un alphabet à un autre

convert_cyr_string()

Retourne une chaîne convertie d'un alphabet cyrillique vers un autre.

Syntaxe	`string convert_cyr_string(string $chaine, string $depuis, string $vers)`
`$chaine`	Chaîne à transformer.
`$depuis`	Alphabet cyrillique de départ.
`$vers`	Alphabet cyrillique voulu.
`retour`	La chaîne de caractères transformée.

Les valeurs de `$depart` et `$vers` sont à prendre parmi les suivantes :

Tableau 7.3 : Codes des alphabets cyrilliques

Code	Désignation
k	koi8-r
w	windows-1251
i	iso8859-5
a	x-cp866
d	x-cp866
m	x-mac-cyrillic

hebrev()

Retourne un texte converti de l'hébreu en texte lisible. Cette fonction n'affecte, hormis les caractères de ponctuation, que les caractères dont le code ASCII est compris entre 224 et 251.

Syntaxe	`string hebrev(string $chaineHebreu [, int $caracteresParLigne]);`
`$chaineHebreu`	Chaîne à transformer.
`$caracteresParLigne`	Nombre de caractères maximum par ligne.
`retour`	La chaîne transformée.

hebrevc()

Cette fonction a le même effet que `hebrev()`, mais transforme, en plus, les caractères `\n` en `<br />\n`.

Syntaxe	`string hebrevc(string $chaineHebreu [, int $caractèresParLigne]);`
`$chaineHebreu`	Chaîne de caractères à transformer.
`$caracteresParLigne`	Nombre de caractères maximum par ligne.
`retour`	La chaîne transformée.

7.5. Manipulation des balises HTML

Une fonction très utile permet de supprimer les balises PHP et HTML pour, par exemple, retirer les balises ajoutées par des visiteurs d'un forum. Elle s'appelle `strip_tags()`.

strip_tags()

Retourne une chaîne de caractères pour laquelle les balises PHP et HTML ont été supprimées.

Syntaxe	`string strip_tags(string $chaine[, string $balisesPermises])`
`$chaine`	Chaîne de caractères à transformer.
`$balisesPermises`	Chaîne composée de la concaténation des balises HTML à ne pas supprimer.
`retour`	La chaîne transformée.

Voici un petit exemple :

```
<?php
$chaine="Du texte avec <b>du gras</b>, <i>de l'italique</i> et du
✂ <u>soulign&eacute;</u>";
echo $chaine."\n<br />";
echo strip_tags($chaine)."\n<br />";
echo strip_tags($chaine,"<b>")."\n<br />";
echo strip_tags($chaine,"<b><u>")."\n<br />";
?>
```

dont voici le résultat :

```
Du texte avec <b>du gras</b>, <i>de l'italique</i> et du <u>soulign&eacute;</u>
<br />Du texte avec du gras, de l'italique et du soulign&eacute;
<br />Du texte avec <b>du gras</b>, de l'italique et du soulign&eacute;
<br />Du texte avec <b>du gras</b>, de l'italique et du <u>soulign&eacute;</u>
<br />
```


REMARQUE

Amélioration depuis PHP 4.3.2
Cette fonction a été améliorée et gère mieux les signes '<' et '>' qui ne sont pas des balises.

get_meta_tags() (ne fonctionne pas sous Windows)

Permet d'extraire les balises d'une chaîne de caractères, et de les mettre dans un tableau.

Syntaxe	`array get_meta_tags(string $nomFichier [, boolean $cheminInclusion])`
`$nomFichier`	Nom du fichier à traiter.
`$cheminInclusion`	`TRUE` si le fichier doit être recherché dans les chemins standard d'inclusion.
`retour`	Tableau associatif ayant pour clés les noms des balises "`meta`" et, pour valeurs, les contenus des attributs "`content`" de ces mêmes balises.

Voici un script d'exemple suivi du fichier de test :

Listing 7.19 : get_meta_tags.php

```
<?php
    $tableau = get_meta_tags("test.html");
    print_r($tableau);
?>
```

Le fichier de test :

Listing 7.20 : test.html

```
<head>
<meta name="author" content="Damien, Laurent, PEM et Thomas">
<meta name="tags" content="Livre PHP">
</head>
```

Et le résultat obtenu :

```
Array
(
    [author] => Damien , Laurent, PEM et Thomas
    [tags] => Livre PHP
)
```

Suppression des espaces

Certaines fonctions permettent d'effacer les espaces superflues en début et/ou fin de chaînes de caractères.

trim()

Retourne une chaîne de caractères sans les espaces (ou autres caractères) de début et de fin.

Syntaxe	`string trim(string $chaine [, string $listeCaracteres])`
`$chaine`	Chaîne de caractères à transformer.
`$listeCaracteres`	Caractères à supprimer (cette option n'existe que depuis la version 4.1.0 de PHP).
`retour`	La chaîne transformée.

Par défaut, voici un tableau des caractères supprimés :

Tableau 7.4 : Caractères supprimés par défaut

Caractère	Code ASCII (en décimal puis hexadécimal)	Description
`\0`	0 (0x00)	Caractère `NUL`.
`\t`	9 (0x09)	Tabulation horizontale.
`\n`	10 (0x0A)	Nouvelle ligne.
`\x0B`	11 (0x0B)	Tabulation verticale.
`\r`	13 (0x0D)	Retour chariot.
`(espace)`	32 (0x20)	Caractère d'espacement.

Voici un script d'exemple :

Listing 7.21 : trim.php

```
<?php
$chaine="\t\t
\t  Le texte important

   \t\t";
echo "Chaîne de départ:\n";
echo $chaine."\n";
echo "Chaîne après avoir utilisé la fonction trim() sans paramètre:\n";
echo trim($chaine)."\n";
echo "Chaîne après avoir utilisé la fonction trim() avec paramètre:\n";
echo trim($chaine,"\t")."\n";
?>
```

Et voici le résultat obtenu (pour qu'il soit lisible, nous avons remplacé manuellement les tabulations par `\t` :

```
Chaîne de départ:
\t\t
\t  Le texte important

   \t\t
Chaîne après avoir utilisé la fonction trim() sans paramètre:
Le texte important
Chaîne après avoir utilisé la fonction trim() avec paramètre:

\t  Le texte important
```

ltrim()

Retourne une chaîne de caractères dans laquelle toutes les espaces (ou autres caractères) de début de chaîne ont été supprimées.

Syntaxe	`string ltrim(string $chaine [, string $listeCaracteres])`
`$chaine`	Chaîne de caractères à transformer.
`$listeCaracteres`	Caractères à supprimer (cette option n'existe que depuis la version 4.1.0 de PHP).
`retour`	La chaîne transformée.

rtrim()

Retourne une chaîne de caractères dans laquelle toutes les espaces de fin de chaîne ont été supprimées.

Syntaxe	`string ltrim(string $chaine[, string $listeCaracteres])`
`$chaine`	Chaîne de caractères à transformer.
`$listeCaracteres`	Caractères à supprimer (cette option n'existe que depuis la version 4.1.0 de PHP).
`retour`	La chaîne transformée.

`rtrim()` possède un alias appelé `chop()`.

Modification de casse

Pour changer la casse des caractères, il existe quatre fonctions `strToUpper()`, `strToLower()`, `ucFirst()` et enfin `ucWords()`.

Un script regroupant ces quatre fonctions sera présenté après le détail de celles-ci.

strToUpper()

Retourne une chaîne dans laquelle tous les caractères ont été mis en majuscules.

Syntaxe	`string strToUpper(string $chaine)`
`$chaine`	Chaîne de caractères à transformer.
`retour`	La chaîne transformée en majuscules.

strToLower()

Retourne une chaîne dans laquelle tous les caractères ont été mis en minuscules.

Syntaxe	`string strToLower(string $chaine)`
`$chaine`	Chaîne de caractères à transformer.
`retour`	La chaîne transformée en minuscules.

ucFirst()

Retourne une chaîne de caractères dans laquelle le premier caractère de la chaîne a été mis en majuscule (sans que les autres ne soient changés).

Syntaxe	`string ucFirst(string $chaine)`
`$chaine`	Chaîne de caractères à transformer.
`retour`	La chaîne transformée.

ucWords()

Retourne une chaîne de caractères dans laquelle le premier caractère de chacun des mots a été mis en majuscule (sans que les autres ne soient changés).

Syntaxe	`string ucWords(string $chaine)`
`$chaine`	Chaîne de caractères à transformer.
`retour`	La chaîne transformée.

Voici quelques lignes de code présentant ces fonctions :

Listing 7.22 : majuscules.php

```
<?php
    $chaine="cette ChaiNe serA transFormee.";
    echo strtolower($chaine)."<br />\n";
    echo strtoupper($chaine)."<br />\n";
    echo ucfirst($chaine)."<br />\n";
    echo ucwords($chaine)."<br />\n";
?>
```

Et voici le résultat obtenu :

```
cette chaine sera transformee.
CETTE CHAINE SERA TRANSFORMEE.
Cette ChaiNe serA transFormee.
Cette ChaiNe SerA TransFormee.
```

7.6. Insertion de motifs

chunk_split()

Retourne une chaîne de caractères dans laquelle un motif (par défaut, un retour à la ligne) a été inséré à espaces réguliers.

Syntaxe	`string chunck_split(string $chaine [, int $pas [, string $separateur]])`
`$chaine`	Chaîne de caractères à transformer.
`$pas`	Nombre de caractères après lesquels insérer le séparateur. (par défaut 73).
`$separateur`	Caractères utilisés pour séparer deux blocs.
`retour`	La chaîne transformée.

Voici un script d'exemple (très simple) :

Listing 7.23 : chunk_split

```
<?php
    echo chunk_split("abcdefghijklmnopqrstuvwxyz0123456789",6,"<br />\n");
    echo chunk_split("a1b2233e34",2,":");
?>
```

Et le résultat obtenu :

```
abcdef
ghijkl
mnopqr
stuvwx
yz0123
456789
a1:b2:23:3e:34:
```

wordWrap()

Permet d'insérer des coupures régulièrement.

Syntaxe	`string wordWrap(string $chaine [, int $largeur [, string $cassure [, bool $coupure]]])`
`$chaine`	Chaîne de caractères à transformer.
`$largeur`	Nombre de caractères après lesquels insérer les caractères de cassure (73 par défaut).
`$cassure`	Chaîne de caractères servant à la cassure (`\n` par défaut).
`$coupure`	Indique si un mot doit être ou non coupé.
`retour`	La chaîne de caractères transformée.

str_pad()

Permet de compléter une chaîne de caractères par un motif.

Syntaxe	`string str_pad(string $chaine, int $longueurFinale [, string $motif [, int $alignement]])`
`$chaine`	Chaîne de caractères à transformer.
`$longueurFinale`	Longueur que fera la chaîne de caractères après transformation.
`$motif`	Motif pour le remplissage (caractère d'espacement par défaut).
`$alignement`	Au choix : `STR_PAD_RIGHT` (par défaut), si le motif doit être répété à droite de la chaîne. STR_PAD_LEFT, si le motif doit être répété à gauche de la chaîne. `STR_PAD_BOTH`, si le motif doit être répété de part et d'autre de la chaîne.
`retour`	La chaîne de caractères transformée.

Voici un script d'exemple très simple :

Listing 7.24 : str_pad.php

```
<?php
    echo "[".str_pad("Cool", 10)."]\n";
    echo "[".str_pad("Cool", 10, " ", STR_PAD_LEFT)."]\n";
    echo "[".str_pad("Cool", 10, " ", STR_PAD_BOTH)."]\n";
    echo "[".str_pad("Cool", 10, "-=", STR_PAD_BOTH)."]\n";
?>
```

Et le résultat obtenu :

```
[Cool      ]
[      Cool]
[   Cool   ]
[-=-Cool-=-]
```


7.7. Fusion et découpe

implode()

Permet, à partir d'un tableau, de reconstituer une chaîne de caractères.

Syntaxe	`string implode([string $entreElements,] array $tableau)`
`$entreElements`	Chaîne de caractères à placer entre deux éléments du tableau. Chaîne vide par défaut depuis PHP 4.3.0.
`$tableau`	Tableau de chaînes de caractères.
`retour`	La chaîne de caractères reconstituée.

Listing 7.25 : explode.php

```
<?php
    print_r(explode("\n","Ceci\nest\nune\nphrase en plusieurs\nlignes"));
    print(implode(" ",
       explode("\n","Ceci\nest\nune\nphrase en plusieurs\nlignes")));
?>
```

Voici le résultat de ce script :

```
Array
(
    [0] => Ceci
    [1] => est
    [2] => une
    [3] => phrase en plusieurs
    [4] => lignes
)
Ceci est une phrase en plusieurs lignes
```

explode()

Permet de retourner un tableau contenant les morceaux de chaînes séparés par un délimiteur défini par le programmeur.

Syntaxe	`array explode(string $separateur, string $chaine [, int $limite])`
`$separateur`	Chaîne de caractères séparant les différents blocs.
`$chaine`	Chaîne de caractères à transformer.
`$limite`	Nombre maximal de blocs, le restant étant mis dans le dernier bloc.
`retour`	Le tableau indexé en question.

strtok()

Permet de parcourir une chaîne morceau par morceau, par appels successifs à la fonction.

Syntaxe	`string strtok(string $chaine, string $separateur)`
`$chaine`	Chaîne de caractères à parcourir.
`$separateur`	Définit les caractères séparant deux morceaux ; n'importe lequel des caractères de séparation est considéré comme une coupure entre deux sous-chaînes.
`retour`	Un des morceaux.

Voici quelques exemples :

Listing 7.26 : strtok.php

```
<?php
//exemple 1
    $sousChaine=strtok("Element 1|Element 2|Element 3","|");
    while ($sousChaine) {
    echo $sousChaine."<br />\n";
    $sousChaine=strtok("|");
    }
//exemple 2
    $sousChaine=strtok("Element 1|Element 2|Element 3"," |");
    while ($sousChaine) {
    echo $sousChaine."<br />\n";
    $sousChaine=strtok(" |");
    }
//exemple 3
    echo strtok("Element 1|Element 2|Element 3","|")."<br />\n";
    echo strtok(" ")."<br />\n";
    echo strtok("n")."<br />\n";
?>
```

Et voici le résultat :

```
Element 1
Element 2
Element 3
Element
1
Element
2
Element
3
Element 1
Element
2|Eleme
```

Autres...

str_repeat()

Permet de répéter une chaîne.

Syntaxe	`string str_repeat(string $chaine, int $nb)`
`$chaine`	Chaîne de caractères à répéter.
`$nb`	Nombre de fois où la chaîne doit être répétée.
`retour`	La chaîne de caractères répétée.

Voici un code d'exemple :

Listing 7.27 : str_repeat.php

```
<?php
    echo str_repeat(":-",20);
    echo str_repeat(":-) ",10);
?>
```

Et le résultat correspondant :

```
:-:-:-:-:-:-:-:-:-:-:-:-:-:-:-:-:-:-:-:-:-:-
:-) :-) :-) :-) :-) :-) :-) :-) :-) :-)
```

strrev()

Retourne une chaîne de caractères dans laquelle l'ordre des caractères a été inversé.

Syntaxe	`string strrev(string $chaine)`
`$chaine`	Chaîne à retourner.
`retour`	Chaîne inversée.

Voici un exemple :

```
<?php
   $chaine="Et se resservir ivresse reste";
   echo $chaine."<br />\n";
   echo strrev($chaine)."<br />\n";

   function estUnPalindrome($chaine) {
       return ($chaine==strrev($chaine)) ? "$chaine est un palindrome" :
          "$chaine n'est pas un palindrome";
   }
```

```
    echo estUnPalindrome("ici")."<br />\n";
    echo estUnPalindrome("chocolat")."<br />\n";
    echo estUnPalindrome("radar")."<br />\n";
    echo estUnPalindrome("rotor")."<br />\n";
    echo estUnPalindrome("voiture")."<br />\n";

?>
```

dont voici le résultat :

```
Et se resservir ivresse reste
etser esservi rivresser es tE
ici est un palindrome
chocolat n'est pas un palindrome
radar est un palindrome
rotor est un palindrome
voiture n'est pas un palindrome
```

str_rot13()

Effectue une permutation circulaire sur les lettres de l'alphabet, chacune des lettres étant décalée de treize places ; ainsi 'a' deviendra 'n'.

Syntaxe	`string str_rot13(string $chaine)`
`$chaine`	Chaîne de caractères à transformer.
`retour`	Chaîne de caractères transformée.

Voici un script d'exemple :

Listing 7.28 : str_rot13.php

```
<?php
    echo str_rot13("abcdefghijklmnopqrstuvwxyz")."\n";
    echo str_rot13("ABCDEFGHIJKLMNOPQRSTUVWXYZ")."\n";
    echo str_rot13(str_rot13("abcdefghijklmnopqrstuvwxyz"))."\n";
?>
```

dont le résultat est :

```
nopqrstuvwxyzabcdefghijklm
NOPQRSTUVWXYZABCDEFGHIJKLM
abcdefghijklmnopqrstuvwxyz
```

Cryptage
Même si cela pourrait ressembler à une fonction de cryptage, il n'en est rien vu que cette fonction est très facilement réversible (il suffit de l'appliquer à elle-même).

Somme de contrôle et cryptage

crc32()

Calcul d'une somme de contrôle sur 32 bits de la chaîne de caractères. Ce calcul est particulièrement utile pour vérifier qu'une chaîne de caractères n'a pas été altérée (suite à une déficience logicielle ou matérielle, ou encore à un acte de malveillance).

Syntaxe	`int crc32(string $chaine)`
`$chaine`	Chaîne dont vous souhaitez la "checksum".
`retour`	*Checksum* sur 32 bits.

Voici un exemple :

Listing 7.29 : crc32.php

```
<?php
    echo crc32("Chaine de caracteres");
    echo "<br />\n";
    echo crc32("Chaine");
    echo "<br />\n";
    echo crc32("Test");
?>
```

dont le résultat serait :

```
169821353
-1820961062
2018365746
```

md5()

Retourne une version cryptée d'une chaîne de caractères basée sur le calcul du md5 (chaîne hexadécimale de 32 caractères).

Syntaxe	`string md5(string $chaine)`
`$chaine`	Chaîne de caractères dont vous souhaitez calculer le md5.
`retour`	md5 de la chaîne.

Listing 7.30 : md5.php

```
<?php
    echo md5("Calcul de md5");
    echo "<br />\n";
    echo md5("Un autre calcul de md5");
?>
```

dont le résultat est :

```
15c620ee8e52f6143383138a079bf54b
05c411550c749adc841fa0f2f9a2cc13
```

crypt()

Retourne une version cryptée d'une chaîne de caractères basée sur la fonction d'encryptage DES.

Syntaxe	`string crypt(string $chaine [, string $sel])`
`$chaine`	Chaîne dont vous souhaitez le cryptage.
`$sel`	Chaîne de deux caractères permettant de calculer la clé (si aucune n'est fournie, PHP se charge d'en créer une aléatoirement). Pour comparer une chaîne à une chaîne cryptée, vous devrez utiliser comme valeurs les deux premiers caractères de la chaîne cryptée (ou la chaîne cryptée entière, sachant que seuls les deux premiers caractères seront pris en compte).
`Retour`	Clé obtenue à partir de la chaîne de caractères et du 'sel'.

Listing 7.31 : crypt.php

```
<?php
    $motDePasseInitiale = "motDePasse";
    $cle = crypt($motDePasseInitiale);
    echo $cle."<br />\n";
    $motDePassePropose = "motDePasse";
    if (crypt($motDePassePropose, $cle) == $cle)
        echo "Le mot de passe est bon";
    else
        echo "Le mot de passe est faux";
?>
```

Le résultat est :

```
t.Gvi9zyHUxp2
Le mot de passe est bon
```

REMARQUE

Cryptologie

Les fonctions `md5()` et `crypt()` sont de véritables fonctions de cryptage. De ce fait, elles ne sont pas réversibles. La comparaison d'une chaîne de caractères avec une autre stockée sous sa forme cryptée doit toujours se faire en cryptant la chaîne fournie, et en comparant le résultat obtenu avec la version stockée (et non en décryptant la version stockée et en comparant le résultat avec la chaîne fournie). Pour la fonction `crypt()`, qui nécessite un paramètre supplémentaire, vous devrez utiliser la version cryptée, comme cela est fait dans l'exemple précédent.

7.8. Expressions régulières

Une expression régulière sert à faire l'analyse lexicale (*parser*) d'une chaîne de caractères. Elle va servir, par exemple, à repérer une valeur dans une chaîne, ou encore à repérer des sous-chaînes particulières. Pour cela, il va falloir définir un motif que l'on appelle 'expression régulière'.

Il existe deux normes pour définir une expression régulière, Perl et POSIX. Entre Perl et POSIX, les différences sont minimes. Si vous connaissez les expressions régulières du monde UNIX, alors vous connaissez déjà la norme POSIX.

Perl

Généralités

La plus simple

La plus simple des expressions régulières est une sous-chaîne de caractères composée de chiffres, de lettres et d'espaces.

L'expression régulière "`morceau à rechercher`" pourra servir à rechercher la sous-chaîne "`morceau à rechercher`" dans une chaîne.

Les métacaractères

Le point

Un caractère très utile est le point "`.`". Dans une expression régulière, il remplace n'importe quel caractère.

Par exemple, "`PHP. est la dernière version`" est une expression regulière pour "PHP3 est la dernière version" ou "PHP4 est la dernière version", mais ne fonctionnera pas pour "PHP 10 est la dernière version", car 10 est sur deux caractères. (Inversement "`PHP.. est la dernière version`" fonctionnera pour la version 10, mais pas pour les versions 3 et 4.)

Utiliser le point tel quel peut poser problème si l'on veut rechercher le caractère point, et uniquement celui-ci. En effet, pour rechercher "`3.14`", il ne faudra pas écrire "`3.14`", sans quoi les expressions "`3F14`", "`3214`", "`3:14`" seront reconnues par cette expression régulière. Pour utiliser le caractère point, il faut le faire précéder du caractère d'échappement: "`\`". Pour rechercher "`3.14`" il faut donc écrire "`3\.14`".

Le point d'interrogation

Le point d'interrogation permet d'indiquer la présence d'au plus une occurrence d'un caractère. Le point d'interrogation est à mettre après le caractère en question.

Voici quelques exemples :

"`chaînes? de caractères`" reconnaîtra "`chaînes de caractères`" et "`chaîne de caractères`".

"`points? et interrogations?`" permettra de reconnaître "`points et interrogations`" "`points et interrogation`", "`point et interrogations`" et "`point et interrogation`".

Pour utiliser le caractère ? en tant que simple caractère (et non comme métacaractère), il faut le faire précéder de '\'.

Le signe plus

Le signe plus (+) permet d'indiquer la présence d'une ou plusieurs occurrences d'un caractère. Le signe plus est à mettre après le caractère en question.

Voici quelques exemples :

"Whaoo+" reconnaîtra "Whaoo", "Whaooo", "Whaoooo" mais pas "Whao".

"coo+l" servira pour reconnaître "cool", "coool", "cooooooool" …mais pas "col".

Pour utiliser le caractère + en tant que simple caractère (et non comme métacaractère), il faut le faire précéder de '\'.

Le signe multiplier

Le signe multiplier (*)permet d'indiquer la présence d'aucune, d'une ou de plusieurs occurrences d'un caractère. Le signe multiplier est à mettre après le caractère en question.

Voici quelques exemples:

"Whao*" reconnaîtra "Wha", "Whao", "Whaoo"…

"cooo*l" servira pour reconnaître "cool", "coool", "cooooooool"…

Pour utiliser le caractère * en tant que simple caractère (et non comme métacaractère), il faut le faire précéder de '\'.

Combinaison de métacaractères

Le point peutêtre combiné avec les signes ?, + ou *. Cela permettra d'indiquer, par exemple, la présence de n'importe quel caractère au moins une fois dans le cas du signe +.

Voici quelques exemples :

"C'est .*bien" servira pour "C'est très bien", "C'est vraiment bien", "C'est rien bien" et même "C'est bien".

Exemples

Voici une série d'exemples. Les expressions régulières peuvent vite devenir complexes à lire ; il suffit de procéder par étapes pour éviter les erreurs :

Tableau 7.5 : Exemples d'expressions régulières

Expression	Définition	Exemples reconnus	Exemples non reconnus
a.z	Caractère 'a' suivi d'un seul caractère suivi de 'z'	a z abz a_z azz	az abcz a_-z
a\.z	Caractère 'a' suivi du caractère '.' suivi de 'z'	a.z	abz

Expression	Définition	Exemples reconnus	Exemples non reconnus
a.+z	Caractère 'a' suivi d'au moins un caractère suivi de 'z'	abz abcz azzz	az
a\.+z	Caractère 'a' suivi d'au moins un caractère '.' suivi de 'z'	a.z a..z a…z	az
a\++	Caractère 'a' suivi d'au moins un caractère '+' suivi de 'z'	a+z a++z a+++z	az
ab?c+d*e	Caractère 'a' suivi d'un ou d'aucun caractère 'b' puis d'un caractère 'c' ou plus, puis d'éventuellement 1 ou plusieurs 'd' puis d'un 'e'	abcde acde abccccccde abce abccccce abcdddde	bcde abcd abde abbcde

Pour affiner la recherche, il est possible de définir certains types de caractères.

Tableau 7.6 : Ensembles de caractères

Notation	Définition
\d	Tout caractère numérique (0, 1, 2, 3, 4, 5, 6, 7, 8, 9).
\D	Tout caractère non numérique.
\w	Tout caractère alphanumérique et le signe souligné '_'.
\W	Tout caractère qui n'est pas alphanumérique ni le caractère souligné '_'.
\s	Tous les caractères d'espacement (espace, tabulation, retour chariot) et tout autre caractère qui n'utiliserait pas d'encre sur une imprimante.
\S	Tous les caractères qui ne sont pas des caractères d'espacement.
\b	Tous les caractères qui entourent un mot (caractères d'espacement, début et fin de ligne, ponctuation).
\B	L'ensemble des caractères qui ne sont pas dans \b.
\nnn	Permet de définir un caractère par son code ASCII en base 8.

Voici une nouvelle série d'exemples illustrant ce que l'on peut définir avec ces ensembles de caractères.

Tableau 7.7 : Exemples

Expression	Définition	Exemples reconnus	Exemples non reconnus
\w+@\w+\.com	Une série de caractères alphanumériques (ou '_') puis le signe '@' puis une autre série de caractères alphanumérique (ou '_') suivi de '.com'.	thomas@toutestfacile .com	thomas@toutestfacile.fr thomas@toutestfacile toto.com
\d\d\d\d\d\	Un nombre à cinq chiffres.	01234 34534	0123 abcde
\b\d\w\D\w\ d\s\W\d	Le début de ligne suivi d'un chiffre puis d'un caractère alphanumérique, puis d'un caractère qui n'est pas un chiffre, puis d'un caractère alphanumérique, puis d'un chiffre, d'un caractère d'espacement, d'un caractère non alphanumérique et enfin d'un chiffre.	0abc1 -2 0_aa1 %2	0a1b1 -2 0abc1 d2

Il est également possible de préciser les nombres minimum et maximum d'occurrences d'un caractère en utilisant une notation entre accolades, les deux valeurs étant séparées par une virgule.

`ab{2,4}c` signifie : un caractère '`a`' suivi de deux à quatre caractères '`b`' puis du caractère '`c`'.

Pour permettre une alternative, il suffit d'utiliser le caractère '`|`'.

`ab|ac` signifie : soit la chaîne '`ab`', soit la chaîne '`ac`' et aucune autre.

Enfin, pour permettre certains caractères ou une certaine plage de caractères, on peut utiliser des crochets.

`a[bcde]f` permettra de reconnaître '`abf`', '`acf`', '`adf`' et '`aef`' mais pas '`abcf`' par exemple.

Pour définir une certaine plage de caractères, il suffit de placer un signe '`-`' entre les caractères délimitant la plage.

`a[b-e]f` est équivalent à l'exemple précédent.

Pour ajouter le signe '-' à la liste des caractères possibles, il suffit de le placer juste avant le crochet fermant.

`a[b-e-]f` permettra de reconnaître '`abf`', '`acf`', '`adf`', '`aef`' et '`a-f`'.

L'utilisation des crochets peut également permettre d'exclure certains caractères en utilisant le caractère d'exclusion `^`.

`[^bc]oule` permettra de reconnaître '`foule`', '`roule`' mais pas '`boule`' ni '`coule`'.

À l'intérieur des crochets, les règles d'échappement ne s'appliquent pas de la même manière ; seuls les caractères `[`, `]` et `\` doivent être précédé du signe `\`.

Tableau 7.8 : D'autres exemples

Expression	Définition	Exemples reconnus	Exemples non reconnus
`[02468]{3,5}`	Un nombre de trois à cinq caractères composé de chiffres pairs.	24804 8602 666	135 1222 20
`[a-z]_[0-9]{3,3}`	Une lettre minuscule suivie du signe '_' puis d'exactement trois chiffres.	a_111 g_753	7_765 4_a33
`[\[\]\\]+`	Une composition de caractères '`[`', '`]`' et '`\`'.	\[]\][]\][[\] \\\]][[[	[a]
`[a-zA-Z0-9._-]+@[a-zA-Z0-9]+\.[a-zA-Z]{2,3}`	Un ou plusieurs caractères alphanumériques ou '`.`', '`_`', '`-`' suivi de '`@`', de un ou plusieurs caractères alphanumériques, d'un point puis de deux ou trois lettres.	livrephp@toutestfacile.com manuel@brazil.br thomas.heute@toutestfacile.com	@toutestfacile.com toto@_.com

Début et fin de ligne

En dehors des crochets, le caractère '`^`' permet de définir le début d'une chaîne. Le caractère '`$`', lui, désigne la fin d'une chaîne.

`abc` reconnaîtra "`abc`", "`dabc`", "`dabce`".

`^abc` reconnaîtra "`abc`" et "`abcd`".

`^abc$` reconnaîtra "`abc`" (uniquement).

Les options

Les expressions régulières Perl sont généralement écrites entre deux slashes '`/`'. Les caractères avant le premier slash et ceux après le dernier permettent de spécifier quelques options.

Le caractère avant le premier slash est sans effet avec les fonctions PHP.

Celui situé après le dernier slash peut être par exemple un :

- '`i`' afin de préciser que la recherche doit être insensible à la casse.
- '`s`' afin que le métacaractère '`.`' concerne également les retours à la ligne.
- D'autres valeurs encore.

Les fonctions PHP

Filtrage par expression régulière

preg_grep()

Retourne un tableau ne contenant que les éléments vérifiant une expression régulière.

Syntaxe	`array preg_grep(string expReg, array $chaines)`
`$expReg`	L'expression régulière filtrante.
`$chaines`	Chaînes de caractères à vérifier.
`retour`	Tableau sans les valeurs ne vérifiant pas l'expression régulière.

Listing 7.32 : preg_grep.php

```
<?php
    $chaines = array(
        "toto@blabla.fr",
        "toto$blabla.fr",
        "toto$blabla.com",
        "toto_titi@blabla.com",
    );
    print_r(preg_grep("/[a-zA-Z0-9._-]+@[a-zA-Z0-9]+\.[a-zA-Z]{2,3}/",
                      $chaines));
?>
```

dont le résultat est :

```
Array
(
    [0] => toto@blabla.fr
    [3] => toto_titi@blabla.com
)
```

Substitution par expression régulière

preg_replace()

Recherche une portion de chaîne de caractères correspondant à une expression régulière et la remplace.

Syntaxe	`mixed preg_replace(mixed $motif, mixed $remplacement, mixed $chaine[, int $limite])`
`$motif`	L'expression régulière à rechercher. Peut être un tableau ; dans ce cas, si `$remplacement` est un tableau, alors les motifs seront remplacés par l'élément de même index de `$remplacement`.
`$remplacement`	Chaîne de substitution ; si celui-ci est une chaîne de caractères, alors tous les motifs trouvés seront remplacés par cette chaîne.
`$chaine`	Chaîne de caractères où effectuer les modifications. Si `$chaine` est un tableau, alors la recherche s'effectue sur tous les éléments du tableau.
`$limite`	Si cet argument est spécifié, alors au plus `$limite` occurrences seront remplacées.
`retour`	`$chaine` modifiée (sera un tableau si `$chaine` en est un).

Voici un code source d'exemple :

Listing 7.33 : preg_replace.php

```
<?php
    echo preg_replace("/\d+/", "...des chiffres...",
                      "En 2000, 1123442 serveurs ...")."<br />\n";
    $chaines=array("En 2000, 1123442 serveurs ...",
                   "12 elephants sur un arbre");
    print_r($chaines)."<br />\n";
    print_r(preg_replace("/\d+/", "...des chiffres...", $chaines));
    $motifs = array("/([a-zA-Z0-9._-]+)@([a-zA-Z0-9]+)\.com/",
                    "/([a-zA-Z0-9._-]+)@([a-zA-Z0-9]+)\.fr/");
    $remplacement = array("Une adresse en .com", "Une adresse en .fr");
    $chaine = "test@toutestfacile.com et test@toutestfacile.fr";
    echo $chaine."<br />\n";
    print_r(preg_replace($motifs, $remplacement, $chaine));
?>
```

dont le résultat est :

```
En ...des chiffres..., ...des chiffres... serveurs ...<br />
Array
(
    [0] => En 2000, 1123442 serveurs ...
    [1] => 12 elephants sur un arbre
)
Array
(
    [0] => En ...des chiffres..., ...des chiffres... serveurs ...
    [1] => ...des chiffres... elephants sur un arbre
)
test@toutestfacile.com et test@toutestfacile.fr<br />
Une adresse en .com et Une adresse en .fr
```

Pour réutiliser un motif capturé en remplacement, il suffit d'y faire appel avec la syntaxe `\\n`, où `n` est la position du motif capturé. Par exemple :

Listing 7.34 : preg_replace2.php

```
<?php
    echo preg_replace("/(\d+)/", "'\\1'", "En 2000, 1123442 serveurs ...").
         "<br />\n";
?>
```

dont le résultat est :

```
En '2000', '1123442' serveurs ...
```


REMARQUE

preg_replace() vs. str_replace()
Bien qu'il soit possible d'utiliser `preg_replace()` à la place de `str_replace()`, il est tout de même préférable, pour des raisons de rapidité, d'utiliser `str_replace()` lorsque le motif recherché s'avère être une chaîne ne faisant intervenir aucun joker. Bien entendu, la différence ne se fera ressentir que si cette fonction est appelée de nombreuses fois.

preg_replace_callback()

Cette fonction est semblable à la précédente, si ce n'est qu'elle permet d'appeler une fonction avec, en paramètre, les motifs capturés.

Syntaxe	`mixed preg_replace_callback(mixed $motif, mixed $fonction, mixed $subject [, int $limite])`
`$motif`	L'expression régulière à rechercher. Peut être un tableau ; dans ce cas, si `$remplacement` est un tableau, alors les motifs seront remplacés par l'élément de même index de `$remplacement`.
`$fonction`	Fonction à appeler qui fera la transformation à partir des motifs capturés.
`$chaine`	Chaîne de caractères où effectuer les modifications. Si `$chaine` est un tableau, alors la recherche s'effectue sur tous les éléments du tableau.
`$limite`	Si cet argument est spécifié, alors au plus `$limite` occurrences seront remplacées.
`retour`	`$chaine` modifiée (sera un tableau si `$chaine` en est un).

Découpe par expression régulière

preg_split()

Permet de casser une chaîne en sous-éléments en précisant le délimiteur par une expression régulière.

Syntaxe	`array preg_split(string $motif, string $chaine [, int $limite [, int $mode]])`
`$motif`	L'expression régulière qui servira de délimiteur.
`$chaine`	Chaîne de caractères à casser.
`$limite`	Nombre limite de sous-chaînes.
`$mode`	Options pouvant être cumulées par `OU` logique (|). *PREG_SPLIT_NO_EMPTY* : seules les chaînes non vides seront retournées. *PREG_SPLIT_DELIM_CAPTURE* : les expressions entre parenthèses entre les délimiteurs de motifs seront aussi capturées.
`retour`	Le tableau indexé des sous-chaînes.

Voici un exemple :

Listing 7.35 : preg_split.php

```
<?php
    print_r(preg_split("/[:;-]/", "element 1:element 2-element 3;element 4"));
?>
```

dont le résultat est :

```
Array
(
    [0] => element 1
    [1] => element 2
    [2] => element 3
    [3] => element 4
)
```

Extraction par expression régulière

preg_match()

Recherche un motif dans une chaîne de caractères.

Syntaxe	`int preg_match(string $motif, string $chaine [, array &$resultat [, int $option])`
`$motif`	Le motif à retrouver.
`$chaine`	Chaîne de caractères dans laquelle rechercher le motif.
`$resultat`	Variable dans laquelle sera copié un tableau indexé dont l'élément d'indice 0 est le motif en entier, l'élément 1 le premier élément capturé (entre parenthèses), l'élément 2 le deuxième, etc.
`$option`	Seul `PREG_OFFSET_CAPTURE` est disponible. Cette option n'est disponible que depuis PHP 4.3.0
`retour`	Le nombre d'éléments capturés, ou `FALSE` si une erreur s'est produite.

Listing 7.36 : preg_match()

```
<?php
    echo preg_match("/([a-zA-Z0-9._-]+)@([a-zA-Z0-9]+)\.([a-zA-Z]{2,3})/",
    "Voici mon adresse: biblephp@exemple.com, elle est bidon",$elements);
    print_r($elements);
?>
```

dont le résultat est :

```
1
Array
(
    [0] => biblephp@exemple.com
    [1] => biblephp
    [2] => exemple
    [3] => com
)
```


preg_match_all()

Recherche un motif dans une chaîne de caractères, et réitère la recherche sur tout le reste de la chaîne de caractères.

Syntaxe	`int preg_match_all(string $motif, string $chaine, array &$resultat [, int $option])`
`$motif`	Le motif à rechercher.
`$chaine`	La chaîne de caractères à rechercher.
`$resultat`	Le tableau où stocker les résultats.
`$option`	Permet entre autres de définir l'ordre dans lequel ranger les résultats. Cela peut être défini par les constantes : `PREG_PATTERN_ORDER` : `$resultat[0]` sera un tableau indexé contenant les motifs en entier, `$resultat[1]` sera un tableau contenant les premiers motifs entre parenthèses.

	PREG_SET_ORDER : $resulat[0] sera un tableau des motifs entre parenthèses de la première sous-chaîne reconnue. PREG_OFFSET_CAPTURE est également disponible depuis PHP 4.3.0, elle peut être combinée à l'une des deux précédentes.
retour	Retourne le nombre de motifs retrouvés, ou FALSE si une erreur s'est produite.

Listing 7.37 : preg_match_all.php

```
<?php
echo preg_match_all("/<(\w*)>([^<]*)<\/\w*>/",
               "<b>Gras</b><i>Italique</i><u>Souligne</u>",
               $resultat)."\n";
print_r($resultat);
echo preg_match_all("/<(\w*)>([^<]*)<\/\w*>/",
               "<b>Gras</b><i>Italique</i><u>Souligne</u>",
               $resultat, PREG_PATTERN_ORDER )."\n";
print_r($resultat);
echo preg_match_all("/<(\w*)>([^<]*)<\/\w*>/",
               "<b>Gras</b><i>Italique</i><u>Souligne</u>",
               $resultat, PREG_SET_ORDER )."\n";
print_r($resultat);
?>
```

Le résultat de ce script est :

```
3
Array
(
    [0] => Array
        (
            [0] => <b>Gras</b>
            [1] => <i>Italique</i>
            [2] => <u>Souligne</u>
        )

    [1] => Array
        (
            [0] => b
            [1] => i
            [2] => u
        )

    [2] => Array
        (
            [0] => Gras
            [1] => Italique
            [2] => Souligne
        )

)
```

```
3
Array
(
    [0] => Array
        (
            [0] => <b>Gras</b>
            [1] => <i>Italique</i>
            [2] => <u>Souligne</u>
        )

    [1] => Array
        (
            [0] => b
            [1] => i
            [2] => u
        )

    [2] => Array
        (
            [0] => Gras
            [1] => Italique
            [2] => Souligne
        )

)
3
Array
(
    [0] => Array
        (
            [0] => <b>Gras</b>
            [1] => b
            [2] => Gras
        )

    [1] => Array
        (
            [0] => <i>Italique</i>
            [1] => i
            [2] => Italique
        )

    [2] => Array
        (
            [0] => <u>Souligne</u>
            [1] => u
            [2] => Souligne
        )

)
```

Divers

preg_quote()

Permet d'échapper les caractères spéciaux (. ,\\, +, * ,?, [, ^,], $, (,), {, }, =, !, <, >, |, :) des expressions régulières.

Syntaxe	`string preg_quote(string $chaine [, string $delimiteur])`
`$chaine`	Chaîne de caractères dont vous souhaitez échapper les caractères spéciaux.
`$delimiteur`	Caractère qui sera également échappé.
`retour`	Une chaîne de caractères dont les caractères ne sont pas des caractères spéciaux.

Voici un court exemple montrant l'importance d'échapper les caractères spéciaux :

Listing 7.38 : preg_quote.php

```
<?php
    $chaine="2*3+4";

    // retournera 0 car l'expression regulière filtre les expressions
    // dont la définition est :
    // "0 ou plus 2 suivi de au moins un 3 et d'un 4"
    echo preg_match("/$chaine/","2*3+4")."\n";

    echo preg_quote($chaine)."\n";
    // Ici les caractères + et = seront échappés.
    echo preg_match("/".preg_quote($chaine)."/","2*3+4");
?>
```

dont le résultat est :

```
0
2\*3\+4
1
```

Posix

Les expressions régulières de POSIX sont très semblables à celles en Perl ; elles sont utilisées sous UNIX pour créer des scripts.

Généralités

La plus simple

La plus simple des expressions régulières est une sous-chaîne de caractères composée de chiffres, de lettres et d'espaces.

L'expression régulière "`morceau à rechercher`" pourra servir à rechercher la sous-chaîne "`morceau à rechercher`" dans une chaîne.

Les métacaractères

Le point

Un caractère très utile est le point "`.`". Dans une expression régulière, il remplace n'importe quel caractère.

Par exemple "`PHP. est la dernière version`" est une expression regulière pour "PHP3 est la dernière version" ou "PHP4 est la dernière version", mais ne fonctionnera pas pour "PHP 10 est la dernière version", car 10 est sur deux caractères. (Inversement "`PHP.. est la dernière version`" fonctionnera pour la version 10, mais pas pour les versions 3 ou 4).

Utiliser le point tel quel peut poser problème si l'on veut rechercher le caractère point, et uniquement celui-ci. En effet, pour rechercher "`3.14`", il ne faudra pas écrire "`3.14`", sans quoi les expressions "`3F14`", "`3214`", "`3:14`" seront reconnues par cette expression régulière.

Pour utiliser le caractère point en tant que simple caractère (et non comme métacaractère), il faut le précéder du caractères d'échappement : "`\`". Pour rechercher "`3.14`" il faut donc écrire "`3\.14`".

Le point d'interrogation

Le point d'interrogation permet d'indiquer la présence d'au plus une occurrence d'un caractère. Le point d'interrogation est à mettre après le caractère en question.

Voici quelques exemples :

"`chaînes? de caractères`" reconnaîtra "`chaînes de caractères`" et "`chaîne de caractères`".

"`points? et interrogations?`" permettra de reconnaître "`points et interrogations`" "`points et interrogation`", "`point et interrogations`" et "`point et interrogation`".

Pour utiliser le caractère `?` en tant que simple caractère (et non comme métacaractère), il faut le faire précéder de '`\`'.

Le signe "plus"

Le signe "plus" (`+`) permet d'indiquer la présence d'une ou plusieurs occurrences d'un caractère. Le signe plus est à mettre après le caractère en question.

Voici quelques exemples :

"`Whaoo+`" reconnaîtra "`Whaoo`", "`Whaooo`", "`Whaoooo`" mais pas "`Whao`".

"`coo+l`" servira pour reconnaître "`cool`", "`coool`", "`cooooooool`" …mais pas "`col`".

Pour utiliser le caractère + en tant que simple caractère (et non comme métacaractère), il faut le faire précéder de '\'.

Le signe multiplier

Le signe multiplier (*)permet d'indiquer la présence d'aucune ou de plusieurs occurrences d'un caractère. Le signe multiplier est à mettre après le caractère en question.

Voici quelques exemples:

"Whao*" reconnaîtra "Wha", "Whao", "Whaoo"…

"cooo*l" servira pour reconnaître "cool", "coool", "cooooooool"…

Pour utiliser le caractère * en tant que simple caractère (et non comme métacaractère), il faut le faire précéder de '\'.

Combinaison de métacaractères

Le point peut être combiné avec les signes ?, + ou *. Cela permettra d'indiquer, par exemple, la présence de n'importe quel caractère au moins une fois dans le cas du signe +.

Voici quelques exemples :

"C'est .*bien" servira pour "C'est très bien", "C'est vraiment bien", "C'est rien bien" et même "C'est bien".

Des exemples

Voici une série d'exemples. Les expressions régulières peuvent vite devenir complexes à lire ; il suffit de procéder par étapes pour éviter les erreurs :

Tableau 7.9 : Exemples

Expression	Définition	Exemples reconnus	Exemples non reconnus
a.z	Caractère 'a' suivi d'un seul caractère suivi de 'z'	a z abz a_z azz	az abcz a_-z
a\.z	Caractère 'a' suivi du caractère '.' suivi de 'z'	a.z	abz
a.+z	Caractère 'a' suivi d'au moins un caractère suivi de 'z'	abz abcz azzz	az
a\.+z	Caractère 'a' suivi d'au moins un caractère '.' suivi de 'z'	a.z a..z a…z	az
a\++	Caractère 'a' suivi d'au moins un caractère '+' suivi de 'z'	a+z a++z a+++z	az

Expression	Définition	Exemples reconnus	Exemples non reconnus
ab?c+d*e	Caractère 'a' suivi d'un ou d'aucun caractère 'b' puis d'un caractère 'c' ou plus, puis d'éventuellement 1 ou plusieurs 'd' puis d'un 'e'	abcde acde abccccccde abce abcccce abcdddde	bcde abcd abde abbcde

Pour affiner la recherche, il est possible de définir certains types de caractères, avec la notation suivante :

Tableau 7.10 : Ensembles de caractères

Notation	Définition
[:digit:]	Tout caractère numérique (0, 1, 2, 3, 4, 5, 6, 7, 8, 9).
[:^digit:]	Tout caractère non numérique.
[:alpha:]	Tout caractère alphabétique.
[:^alpha:]	Tout caractère non alphabétique.
[:alnum:]	Tout caractère alphanumérique.
[:^alnum:]	Tout caractère non alphanumérique.
[:ascii:]	Tout caractère alphanumérique.
[:^ascii:]	Tout caractère non alphanumérique.
[:lower:]	Tout caractère alphabétique en minuscule.
[:upper:]	Tout caractère alphabétique en majuscule.
[:print:]	Tout caractère imprimable.
[:word:]	Tout caractère alphanumérique en minuscule et le signe souligné _.
[:^word]	Tout caractère qui n'est pas alphanumérique en minuscule ni le caractère souligné '_'.
[:space:]	Tous les caractères d'espacement (espace, tabulation, retour chariot) et tout autre caractère qui n'utiliserait pas d'encre sur une imprimante.
[:^space:]	Tous les caractères qui ne sont pas des caractères d'espacement.
[:punct:]	Tout caractère de ponctuation.
[:xdigit:]	Tout caractère hexadécimal ([0-9a-f]).
[:cntrl:]	Tout caractère de contrôle.

Voici une nouvelle série d'exemples illustrant ce que l'on peut définir avec ces ensembles de caractères.

Tableau 7.11 : Exemples

Expression	Définition	Exemples reconnus	Exemples non reconnus
`[[:word:]]+@[[:word:]]+\.com`	Une série de caractères alphanumériques (ou '_') puis le signe '@' puis une autre série de caractères alphanumérique (ou '_') suivi de '.com'.	thomas@toutestfacile.com	thomas@toutestfacile.fr toto.com

`[:digit:][:word:]`
`[:^digit:][:word:]`
`[:digit:][:space:]`
`[:^word:][:digit:]`
	Un chiffre suivi d'un caractère alphanumérique puis d'un caractère qui n'est pas un chiffre puis un caractère alphanumérique puis un chiffre, un caractère d'espacement, d'un caractère non alphanumérique et enfin un chiffre	0abc1 -2
		0_aa1 %2
			0a1b1 -2
			0abc1 d2

Il est également possible de préciser les nombres minimum et maximum d'occurrences d'un caractère en utilisant une notation entre accolades, les deux valeurs étant séparées par une virgule.

`ab{2,4}c` signifie : un caractère '`a`' suivi de deux à quatre caractères '`b`' puis du caractère '`c`'.

Pour permettre une alternative, il suffit d'utiliser le caractère '`|`'.

`ab|ac` signifie : soit la chaîne '`ab`', soit la chaîne '`ac`' et aucune autre.

Enfin, pour permettre certains caractères ou une certaine plage de caractères, on peut utiliser des crochets.

`a[bcde]f` permettra de reconnaître '`abf`', '`acf`', '`adf`' et '`aef`', mais pas '`abcf`' par exemple.

Pour définir une certaine plage de caractères, il suffit de placer un signe '`-`' entre les caractères délimitant la plage.

`a[b-e]f` est identique à l'exemple précédent.

Pour ajouter le signe '`-`' à la liste des caractères, il suffit de le placer juste avant le crochet fermant.

`a[b-e-]f` permettra de reconnaître '`abf`', '`acf`', '`adf`', '`aef`' et '`a-f`'.

L'utilisation des crochets peut également permettre d'exclure certains caractères, en utilisant le caractère d'exclusion '`^`'.

`[^bc]oule` permettra de reconnaître '`foule`', '`roule`' mais pas '`boule`' ni '`coule'`.

À l'intérieur des crochets, les règles d'échappement ne s'appliquent pas de la même manière ; seuls les caractères `[`, `]` et `\` doivent être précédés du signe `\`.

Tableau 7.12 : D'autres exemples

Expression	Définition	Exemples reconnus	Exemples non reconnus
`[02468]{3,5}`	Un nombre de trois à cinq caractères composés de chiffres pairs.	24804 8602 666	135 1222 20
`[a-z]_[0-9]{3,3}`	Une lettre minuscule suivie du signe '_' puis d'exactement trois chiffres.	a_111 g_753	7_765 4_a33
`[\[\]\\]+`	Une composition de caractères '`[`', '`]`' et '`\`'.	\[]\][]\][[\] \\\]][[[	[a]
`[a-zA-Z0-9._-]+@[a-zA-Z0-9]+\.[a-zA-Z]{2,3}`	Un ou plusieurs caractères alphanumériques ou '`.`', '`_`', '`-`' suivi de '`@`', de un ou plusieurs caractères alphanumériques, d'un point puis de deux ou trois lettres.	livrephp@toutestfacile.com manuel@brazil.br thomas.heute@toutestfacile.com	@toutestfacile.com toto@_.com

Début et fin de ligne

En dehors des crochets, le caractère '`^`' permet de définir le début d'une chaîne. Le caractère '`$`' désigne la fin d'une chaîne.

`abc` reconnaîtra "`abc`", "`dabc`", "`dabce`".

`^abc` reconnaîtra "`abc`" et "`abcd`".

`^abc$` reconnaîtra "`abc`" (uniquement).

Les fonctions PHP

Substitution par expression régulière

ereg_replace()

Permet de remplacer une partie de chaîne de caractères par une autre.

Syntaxe	`string ereg_replace(string $expression, string $remplacement, string $chaine)`
`$expression`	Expression régulière qui correspond à la partie à remplacer.
`$remplacement`	Chaîne de substitution qui peut récupérer les motifs capturés par `\\n`, où `n` est le numéro du motif.

`$chaine`	Chaîne dans laquelle remplacer une partie.
`retour`	La chaîne de caractères modifiée.

Voici un script d'exemple :

```
<?php
    echo ereg_replace("([[:alpha:]]+)@([[:alpha:]]+)\.com",
        "Première partie de l'adresse:\\1 Domaine:\\2",
        "webmaster@toutestfacile.com");
?>
```

dont le résultat est :

```
Première partie de l'adresse:webmaster Domaine:toutestfacile
```

ereg_replace() vs. str_replace()

Bien qu'il soit possible d'utiliser `ereg_replace()` à la place de `str_replace()`, il est tout de même préférable, pour des raisons de rapidité, d'utiliser `str_replace()` lorsque le motif recherché s'avère être une chaîne ne faisant intervenir aucun joker. Bien entendu, la différence ne se fera ressentir que si cette fonction est appelée de nombreuses fois.

eregi_replace()

Permet de remplacer une partie de chaîne de caractères par une autre expression régulière insensible à la casse.

Syntaxe	`string eregi_replace(string $expression, string $remplacement, string $chaine)`
`$expression`	Expression régulière qui correspond à la partie à remplacer.
`$remplacement`	Chaîne de substitution qui peut récupérer les motifs capturés par `\\n`, où `n` est le numéro du motif.
`$chaine`	Chaîne dans laquelle remplacer une partie.
`retour`	La chaîne de caractères modifiée.

Voici un script d'exemple :

Listing 7.39 : eregi_replace.php

```
<?php
    echo ereg_replace("([[:alpha:]]+)@([[:alpha:]]+)\.com",
        "Première partie de l'adresse:\\1 Domaine:\\2",
        "webmaster@toutestfacile.Com");
?>
```

dont le résultat est :

```
Première partie de l'adresse:webmaster Domaine:toutestfacile
```

Extraction et comparaison par expression régulière

ereg()

Permet de retourner les motifs capturés d'une chaîne de caractères ou, tout simplement, de vérifier si cette dernière satisfait à une expression régulière.

Syntaxe	`int ereg(string $expression, string $chaine [, array &$resultat])`
`$expression`	Expression régulière à vérifier.
`$chaine`	Chaîne dont on veut vérifier qu'elle correspond à l'expression régulière.
`$resultat`	Les motifs capturés sont stockés dans ce tableau.
`retour`	`TRUE` si la chaîne vérifie l'expression régulière, `FALSE` sinon.

Voici un script d'exemple :

Listing 7.40 : ereg.php

```
<?php
    echo (ereg("([[:alpha:]]+)@([[:alpha:]]+)\.com",
        "webmaster@toutestfacile.com")) ? "Ca matche" : "Ca matche pas";
    echo "<br />\n";
    echo (ereg("([[:alpha:]]+)@([[:alpha:]]+)\.com",
        "webmaster@toutestfacile.COM")) ? "Ca matche" : "Ca matche pas";
?>
```

dont le résultat est :

```
Ca matche
Ca matche pas
```

eregi()

Permet de retourner les motifs capturés d'une chaîne de caractères ou, tout simplement, de vérifier si cette dernière satisfait à une expression régulière, sans tenir compte de la casse (les minuscules et majuscules sont confondues).

Syntaxe	`int eregi(string $expression, string $chaine[, array $resultat])`
`$expression`	Expression régulière à vérifier.

`$chaine`	Chaîne dont vous souhaitez extraire des motifs, ou vérifier qu'elle correspond à l'expression régulière.
`$resultat`	Les motifs capturés sont stockés dans ce tableau.
`retour`	`TRUE` si la chaîne vérifie l'expression régulière, `FALSE` sinon.

Voici un script d'exemple :

Listing 7.41 : eregi.php

```
<?php
    echo (eregi("([[:alpha:]]+)@([[:alpha:]]+)\.com",
        "webmaster@toutestfacile.com")) ? "Ca matche" : "Ca matche pas";
    echo "<br />\n";
    echo (eregi("([[:alpha:]]+)@([[:alpha:]]+)\.com",
        "webmaster@toutestfacile.COM")) ? "Ca matche" : "Ca matche pas";
?>
```

dont le résultat est :

```
Ca matche
Ca matche
```

Découpe par expression régulière

split()

Permet de découper une chaîne de caractères en morceaux selon une expression régulière.

Syntaxe	`array split(string $expression, string $chaine [, int $limite])`
`$expression`	Expression régulière du séparateur.
`$chaine`	Chaîne à découper.
`$limite`	Nombre d'éléments maximum du tableau.
`retour`	Un tableau des différentes parties séparées par l'expression régulière.

Encore un petit exemple d'application :

Listing 7.42 : split.php

```
<?php
    print_r(split("[:;,.]",
            "Comment,separer;des.elements,d'une:chaine"));
?>
```

qui produira :

```
Array
(
    [0] => Comment
    [1] => separer
    [2] => des
    [3] => elements
    [4] => d'une
    [5] => chaine
)
```

spliti()

Permet de découper une chaîne de caractères en morceaux, l'expression régulière ignorant la casse des caractères.

Syntaxe	`array split(string $expression, string $chaine[, int $limite])`
`$expression`	Expression régulière du séparateur.
`$chaine`	Chaîne à découper.
`$limite`	Nombre d'éléments maximum du tableau.
`retour`	Un tableau des différentes parties séparées par l'expression régulière.

Divers

sql_regcase()

Permet de créer une expression régulière négligeant la casse.

Syntaxe	`string sql_regcase(string $chaine)`
`$chaine`	Chaîne de caractères.
`retour`	Une expression régulière.

Encore un exemple :

Listing 7.43 : sql_regcase.php

```
<?php
    echo sql_regcase("C'est Cool.");
?>
```

Et son résultat :

```
[Cc]'[Ee][Ss][Tt] [Cc][Oo][Oo][Ll].
```

7.9. Adapter le texte à la langue du visiteur

Si vous souhaitez proposer le même site web à des visiteurs provenant de différents horizons linguistiques, vous devrez trouver une solution vous permettant d'adapter le texte en fonction de la langue choisie. Une des solutions consiste à utiliser la bibliothèque `gettext`. Malheureusement, sans que nous puissions véritablement déterminer l'origine du problème, il se trouve que cette bibliothèque n'a correctement fonctionné que dans un des trois environnements testés. Nous ne nous étendrons pas sur ce sujet, d'autant qu'il est très facile d'arriver au même résultat sans avoir à utiliser la moindre bibliothèque (ce qui garantit un fonctionnement dans n'importe quel environnement).

En effet, il vous suffit de créer un script par langue que vous souhaitez proposer, chacun de ces scripts définissant un tableau associatif ayant pour clés les identifiants de message (qui pourraient être les versions françaises du message) et pour valeurs les traductions.

Ce qui donne, par exemple :

Listing 7.44 : lang_en_inc.php

```
<?php
    $msg["titre"]         = "Welcome! This is our acme website";
    $msg["Bonjour"]       = "Hi";
    $msg["Quoi de neuf?"] = "What's up?";
?>
```

Listing 7.45 : lang_fr_inc.php

```
<?php
    $msg["titre"]         = "Bienvenue! C'est notre super site";
    $msg["Bonjour"]       = "Bonjour";
    $msg["Quoi de neuf?"] = "Quoi de neuf?";
?>
```

Il suffira alors d'inclure le fichier correspondant à la langue choisie, et de faire appel aux valeurs du tableau pour chaque affichage. Comme dans l'exemple suivant :

Listing 7.46 : lang_accueil.php

```
<?php
    switch ($_GET["lang"]) {
        case "en" :
        {
            include("lang_en_inc.php");
            break;
        }
        case "fr" :
        default :
        {
            include("lang_fr_inc.php");
            break;
        }
    }
```

```
?>
<html>
<head>
<title><?php echo $msg["titre"]; ?></title>
</head>
<body>
<h1><?php echo $msg["Bonjour"]; ?></h1>
<?php echo $msg["Quoi de neuf?"]; ?>
</body>
</html>
```

Ainsi, l'appel `lang_accueil.php?lang=fr` (ou tout autre valeur de `lang` différente de "en") retournera :

```
Bonjour
Quoi de neuf?
```

avec pour titre :

```
"Bonjour! C'est notre super site"
```

Alors que l'appel `lang_accueil.php?lang=en` retournera :

```
Hi
What's up?
```

avec pour titre :

```
Welcome! This is our acme website
```

Même si votre site n'est qu'en français, cette méthode peut présenter des avantages, puisqu'en faisant ainsi, tous les textes sont centralisés dans un unique fichier (ce qui ne peut que simplifier les mises à jour).

Détection automatique de la langue préférée

Si vous souhaitez déterminer automatiquement la langue préférée de votre visiteur, vous pourrez faire appel à la variable externe `$_SERVER["HTTP_ACCEPT_LANGUAGE"]`. Celle-ci contient une liste des codes pays (sur deux lettres) séparés par des virgules et dans l'ordre de préférence. Vous pourrez alors déterminer la langue préférée supportée par votre site (ici, "en" et "fr") grâce au script suivant :

```
<?php
    function languePreferree() {
        $langs = explode(",", $_SERVER["HTTP_ACCEPT_LANGUAGE"]);
        for ($i=0; (($i<count($langs))&&(!isset($lang))); $i++) {
            if (in_array($langs[$i], array("en", "fr"))) $lang = $langs[$i];
        }
        return $lang;
    }
?>
```

Chapitre 8

La gestion des dates et des calendriers

8.1. Les fonctions de date et heure

Le langage PHP offre des fonctions similaires à ce que l'on peut rencontrer en C/C++. La plupart de ces fonctions s'appuient sur une date définie par un entier correspondant au nombre de secondes écoulées depuis le 1er janvier 1970 à 00:00:00 (aussi appelé *epoch*). Une date ainsi définie sera également appelée *timestamp UNIX*.


REMARQUE

Jusqu'en 2037…
Ce timestamp UNIX n'est valable que dans la plage de temps comprise entre le 1er janvier 1970 et 2037.

Récupérer une date au "format informatique"

Ainsi, généralement, la première opération à réaliser lorsque l'on souhaite manipuler des dates consiste à récupérer la date voulue au format timestamp UNIX. Pour cela, nous disposons des fonctions `time()`, `mktime()` et `strtotime()`.

time()

Retourne la date courante.

Syntaxe	`int time(void)`
`retour`	Nombre de secondes écoulées depuis le 1er janvier 1970.

mktime()

Retourne le timestamp de la date spécifiée.

Syntaxe	`int mktime(int $heure, int $minute, int $seconde, int $mois, int $jour, int $annee)`
`$heure`	L'heure.
`$minute`	Les minutes.
`$seconde`	Les secondes.
`$mois`	Le mois.
`$jour`	Le jour.
`$annee`	L'année.
`retour`	Nombre de secondes écoulées entre le 1er janvier 1970 et la date spécifiée.

strtotime()

Retourne le timestamp d'une date spécifiée sous la forme d'une chaîne de caractères et exprimée en anglais.

Syntaxe	`int strtotime(string $date)`
`$date`	Date exprimée en anglais.
`retour`	Nombre de secondes écoulées entre le 1er janvier 1970 et la date spécifiée.

Listing 8.1 : datetime_01.php

```
<?php
echo "Le timestamp actuel est ".time()."<br />";
echo "Le timestamp pour la date du 10/01/2001 18:15 est ";
echo mktime(18, 15, 0, 1, 10, 2001)."<br />";
echo "Le timestamp pour la date du 10 Janvier 2001 18:15 est ";
echo strtotime("10 January 2001 18:15")."<br />";
echo "Le timestamp pour la date du 2001-01-10 18:15 est ";
echo strtotime("2001-01-10 18:15")."<br />";
?>
```

retournera :

```
Le timestamp actuel est (valeur variable)
Le timestamp pour la date du 10/01/2001 18:15 est 979146900
Le timestamp pour la date du 10 Janvier 2001 18:15 est 979146900
Le timestamp pour la date du 2001-01-10 18:15 est 979146900
```


REMARQUE

Dates au format SQL
Comme vous pouvez le constater, `strtotime()` permet de traiter des dates issues d'une base de données (i.e. au format `AAAA-MM-JJ hh:mm:ss`).

Effectuer des opérations sur les dates

Lorsque vous avez une date au format timestamp UNIX (issue, par exemple, de la fonction `time()`), vous pouvez ajouter/soustraire des secondes, des minutes, des heures, des jours simplement en ajoutant/soustrayant le nombre de secondes correspondant.

Listing 8.2 : datetime_02a.php

```
<?php
echo "Le timestamp d'hier à la même heure est ".(time()-24*3600)."<br />";
?>
```

Si, en revanche, vous voulez ajouter/soustraire des mois ou des années, vous devrez décomposer la date exprimée en secondes depuis le 1er janvier 1970 en jours, mois, années, heures, minutes et secondes. Pour cela, vous pouvez faire appel à la fonction `getDate()`.

getDate()

Retourne les différents champs d'une date (ou de la date courante).

Syntaxe	`array getDate([int $date])`
`$date`	Date exprimée en nombre de secondes depuis 1970 (par défaut, la date courante).
`retour`	Tableau associatif contenant (entre autres) les clés : "`mday`" jour. "`mon`" mois. "`year`" année. "`hours`" heure. "`minutes`" minutes. "`seconds`" secondes.

Listing 8.3 : datetime_02b.php

```
<?php
    $tableauDate = getdate();

    // Calcul de la date courante - 6 mois
    $tableauDate["mon"] = $tableauDate["mon"] - 6;

    if ($tableauDate["mon"] < 1) {
        $tableauDate["mon"] = $tableauDate["mon"] + 12;
        $tableauDate["year"] = $tableauDate["year"] - 1;
    }

    echo "Il y a 6 mois, nous étions le ";
    echo $tableauDate["mday"]."/".$tableauDate["mon"];
    echo "/".$tableauDate["year"];
    echo "<br />";
?>
```

Afficher des dates

Il existe différentes fonctions d'affichage (ou, plus exactement, de représentation) des dates, mais toutes s'appuient sur une date exprimée en secondes depuis le 1er janvier 1970.

Parmi elles, vous trouverez les fonctions `date()` et `strftime()`.

date()

Représente une date (ou date courante) selon le format spécifié.

Syntaxe	`string date(string $format [, ind $date])`
`$format`	Chaîne précisant le format de représentation de la date en s'appuyant sur les clés présentées ci-après.
`$date`	Date exprimée en nombre de secondes depuis 1970 (par défaut, date courante).
`retour`	Chaîne de caractères représentant la date.

Dans le cadre de la fonction `date()`, les clés sont :

Tableau 8.1 : Les différentes clés pour représenter un format de date avec date()

Clé	Signification
Concernant les jours	
l	Pour le jour en toutes lettres (en anglais).
D	Pour les trois premières lettres du jour (en anglais).
w	Pour le jour de la semaine (0 = Dimanche, ..., 6 = Samedi).
d	Pour le jour du mois, sur deux chiffres.
j	Pour le jour du mois (sur un ou deux chiffres).
S	Pour le suffixe sur deux lettres (en anglais) du nombre indiquant le jour.
z	Pour le numéro du jour dans l'année (0 = 1er Janvier).
Concernant les mois	
F	Pour le mois en toutes lettres (en anglais).
M	Pour les premières lettres du mois (en anglais).
m	Pour le mois, sur deux chiffres.
n	Pour le mois (sur un ou deux chiffres).
t	Pour le nombre de jours dans le mois.
Concernant les années	
Y	Pour l'année, sur quatre chiffres.
y	Pour l'année, sur deux chiffres.
L	Pour "1" si l'année est bissextile, "0" sinon.
Concernant les heures	
g	Pour l'heure sur 12 heures (sur un ou deux chiffres).
G	Pour l'heure sur 24 heures (sur un ou deux chiffres).
h	Pour l'heure sur 12 heures, sur deux chiffres.

Clé	Signification
H	Pour l'heure sur 24 heures, sur deux chiffres.
a	Pour "am" ou "pm".
A	Pour "AM" ou "PM".
I	Pour "1" en heure d'été et "0" en heure d'hiver.
T	Pour le fuseau horaire.
Z	Pour le décalage horaire GMT en secondes.
B	Pour l'heure internet Swatch.
Concernant les minutes	
i	Pour les minutes, sur deux chiffres.
Concernant les secondes	
s	Pour les secondes, sur 2 chiffres.
Concernant la date dans son ensemble	
r	Pour la date au format RFC 822 (Tue, 22 Jan 2002 11:09:42 +0100).
U	Retourne simplement le timestamp.

L'exemple :

Listing 8.4 : datetime_03a.php

```
<?php
    echo "Nous sommes le ".date("l j F Y")." ";
    echo "Il est ".date("H:i:s")."<br />";
?>
```

pourra retourner quelque chose comme :

```
Nous sommes le Tuesday 9 July 2002
Il est 22:39:06
```

La fonction `date()` propose donc de nombreuses possibilités de formatage d'une date, mais nous lui préférons la fonction `strftime()` qui, elle, tient compte de la langue locale lors de la restitution d'informations en toutes lettres.

strftime()

Représente une date (ou la date courante) selon le format spécifié.

Syntaxe	`string strftime(string $format [, int $date])`
`$format`	Chaîne précisant le format de représentation de la date en s'appuyant sur les clés présentées ci-après.

`$date`	Date exprimée en nombre de secondes depuis 1970 (par défaut, la date courante).
`retour`	Chaîne de caractères représentant la date.

Dans le cadre de la fonction `strftime()`, les clés sont :

Tableau 8.2 : Les différentes clés pour représenter un format de date avec strftime()

Clé	Signification
Concernant le jour	
%A	Pour le jour en toutes lettres (dans la langue locale).
%a	Pour les trois premières lettres du jour (dans la langue locale).
%d	Pour le jour du mois, sur deux chiffres.
%e (*)	Pour le jour du mois (sur un ou deux chiffres).
%j	Pour le numéro du jour dans l'année, sur trois chiffres (1 = 1er janvier).
%u (*)	Pour le jour de la semaine (1 = lundi, ..., 7 = dimanche).
%w	Pour le jour de la semaine (0 = dimanche, ..., 6 = samedi).
Concernant le mois	
%B	Pour le mois en toutes lettres (dans la langue locale).
%b	Pour les trois premières lettres du mois (dans la langue locale).
%h (*)	Comme %b.
%m	Pour le mois; sur deux chiffres.
Concernant l'année	
%y	Pour l'année, sur deux chiffres.
%Y	Pour l'année, sur quatre chiffres.
Concernant l'heure	
%H	Pour l'heure sur 24 heures, sur deux chiffres.
%I	Pour l'heure sur 12 heures, sur deux chiffres.
%p	Pour AM ou PM.
%Z	Pour le fuseau horaire.
Concernant les minutes	
%M	Pour les minutes, sur deux chiffres.
Concernant les secondes	
%S	Pour les secondes, sur deux chiffres.
Concernant la semaine	
%U	Pour le numéro de la semaine (1 = semaine du 1er dimanche de l'année).
%V (*)	Pour le numéro de la semaine (ISO 8601:1988), sur deux chiffres.

Clé	Signification
%W	Pour le numéro de la semaine (1 = semaine du 1er lundi de l'année).
Concernant le siècle	
%C (*)	Pour le siècle, sur deux chiffres.
Concernant la date dans son ensemble	
%D (*)	Pour la date au format %m/%d/%y.
%r (*)	Pour l'heure au format %I:%M:%S %p.
%R (*)	Pour l'heure au format %H:%M.
%T (*)	Pour l'heure au format %H:%M:%S.
%c	Pour l'affichage traditionnel de la date dans la langue locale (ex. : JJ/MM/AA hh:mm:ss en France).
%x	Pour l'affichage traditionnel de la date dans la langue locale, sans l'heure (ex. : JJ/MM/AA en France).
%X	Pour l'affichage traditionnel de l'heure dans la langue locale (ex. : hh:mm:ss en France).
Autre	
%n	Pour insérer un retour à la ligne.
%t	Pour insérer une tabulation.
%%	Pour afficher un pourcentage.

L'exemple :

Listing 8.5 : datetime_03b.php

```
<?php
        setLocale(LC_TIME, "fr");
    echo "Nous sommes le ".strftime("%A %d %B %Y")."<br />";
    echo "Il est ".strftime("%H:%M:%S")."<br />";
?>
```

retournera quelque chose comme :

```
Nous sommes le mardi 09 juillet 2002
Il est 22:51:06
```

Portabilité

Malheureusement, il est possible que votre bibliothèque PHP ait été compilée avec une bibliothèque C ne supportant pas toutes ces options. Les clés marquées d'un astérisque ne sont pas disponibles dans les versions Windows, alors qu'elles le sont parfaitement sous Linux.

Les heures GMT

Les fonctions `mktime()`, `date()` et `strftime()` se déclinent en `gmmktime()`, `gmdate()` et `gmstrftime()`.

Ces fonctions s'utilisent exactement de la même façon, mais :

- `gmmktime()` retourne le timestamp UNIX local (tenant compte du décalage horaire) d'une date donnée en heures GMT.
- `gmdate()` et `gmstrftime()` retournent les valeurs GMT pour des dates exprimées localement (tenant compte du décalage horaire).

Listing 8.6 : datetime_04.php

```
<?php
    echo "Le 1/1/2003 11:00:00 GMT donne ";
    echo strftime("%H:%M:%S",gmmktime(11,0,0,1,1,2003));
    echo "localement <br />";

    echo strftime("A %H:%M:%S ",mktime(12,0,0,1,1,2003));
    echo "localement<br />";

    echo gmstrftime("Il est %H:%M:%S",mktime(12,0,0,1,1,2003));
    echo " GMT (d'après gmstrftime).<br />";

    echo "Il est ".gmdate("H:i:s",mktime(12,0,0,1,1,2003));
    echo " GMT (d'après gmdate).<br />";
?>
```

affichera :

```
Le 1/1/2003 11:00:00 GMT donne 12:00:00localement
A 12:00:00 localement
Il est 11:00:00 GMT (d'après gmstrftime).
Il est 11:00:00 GMT (d'après gmdate).
```

Les microsecondes

PHP dispose de deux fonctions permettant de récupérer l'heure à la microseconde près. Il s'agit des fonctions `getTimeOfDay()` et `microtime()`.

getTimeOfDay()

Retourne les champs de la date courante, y compris les microsecondes.

Syntaxe	`array getTimeOfDay(void)`
`retour`	Tableau associatif contenant (entre autres) les clés : "`sec`" contenant le timestamp UNIX. "`usec`" contenant les microsecondes.

microtime()

Retourne une chaîne de caractères contenant les microsecondes de la date courante.

Syntaxe	`string microtime(void)`
`retour`	Chaîne de caractères contenant les microsecondes (0.xxxxxx), une espace puis le timestamp UNIX.

Listing 8.7 : datetime_05.php

```
<?php
    $top0=getTimeOfDay();
    echo "Temps d'execution avec getTimeOfDay<br />";
    $top1=getTimeOfDay();

    $diff=($top1["usec"]+$top1["sec"]*1E6) -
          ($top0["usec"]+$top0["sec"]*1E6);

    echo "Temps = $diff micro-secondes<br />";

    list($usec0,$sec0)=explode(" ",microtime());
    echo "Temps d'execution avec microtime<br />";
    list($usec1,$sec1)=explode(" ",microtime());

    $diff=($sec1+$usec1) - ($sec0+$usec0);

    echo "Temps = $diff secondes<br />";

?>
```

Autres fonctions

Outre les fonctions permettant de récupérer ou de formater des dates, heures et microsecondes, PHP propose la fonction `checkDate()` qui permet de contrôler la validité d'une date (en tenant compte notamment des années bissextiles).

checkDate()

Teste la validité de la date.

Syntaxe	`boolean checkDate(int $mois, int $jour, int $annee)`
`$mois`	Le mois.
`$jour`	Le jour.
`$annee`	L'année.
`retour`	`TRUE` si la date existe, `FALSE` sinon.

La fonction `localTime()` qui permet de récupérer les différents champs d'une date n'est pas présentée en détail ici, puisque ses fonctionnalités se retrouvent dans les fonctions présentées précédemment.

8.2. Les dates et calendriers particuliers

Vous trouverez également, avec le langage PHP, quelques fonctions vous permettant de déterminer, pour chaque année, des dates particulières (en l'occurrence celle de Pâques), mais aussi des fonctions permettant de convertir des dates d'un calendrier à l'autre (grégorien, julien, juif, républicain).

Pour cela, vous devez faire appel à la bibliothèque `Calendar`.

Installation

Sous Windows

Que ce soit avec l'archive de PHP Group ou avec EasyPHP, la bibliothèque `Calendar` est activée par défaut.

Sous Linux

L'utilisation de cette bibliothèque nécessite d'avoir compilé PHP avec l'option `--enable-calendar`.

Vous pouvez vous reporter au chapitre "Prise en main" pour plus de détails sur la compilation de PHP.

Vérification

Vous pouvez vérifier que le support de la bibliothèque `Calendar` est effectif en appelant un script contenant simplement `<?php phpinfo(); ?>` et qui doit afficher :

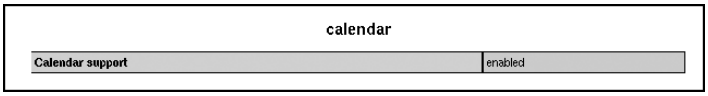


Figure 8.1 : *phpinfo()*

Pâques

Pour connaître la date correspondant à Pâques pour une année donnée, il suffit de faire appel à la fonction `easter_date()`.

easter_date()

Retourne la date de Pâques pour l'année donnée (ou l'année courante).

Syntaxe	`int easter_date([int $annee])`
`$annee`	Année considérée (par défaut, année courante) entre 1970 et 2037.
`retour`	Timestamp UNIX de Pâques.

Si, toutefois, vous souhaitez effectuer la même opération pour une année située en dehors de la plage de validité du timestamp UNIX (i.e. avant le 1er janvier 1970 ou après 2037), vous devez faire appel à `easter_days()`.

easter_days()

Retourne le nombre de jours entre le 21 mars et Pâques pour l'année donnée (ou l'année courante).

Syntaxe	`int easter_days([int $annee])`
`$annee`	Année considérée (par défaut, année courante).
`retour`	Nombre de jours.

Listing 8.8 : calendar_01.php

```
<?php
        setLocale(LC_TIME, "fr");
    echo strftime("Cette année, Pâques tombe le %A %d %B<br />",
                  easter_date());
    echo strftime("En 2001, Pâques était le %A %d %B<br />",
                  easter_date(2001));

    // Pour les années précédant 1970 ou suivant 2037
    // utiliser easter_days
    // La date de référence est le 21 Mars
    $jour   = 21;
    $mois   = 3;
    $annee  = 1969;
    $nbJour = easter_days($annee);
    if ($nbJour > 10) {
        $mois++;
        $jour = $nbJour - 10;
    } else {
        $jour += $nbJour;
    }
    echo "En $annee, Pâques c'est le $jour/$mois<br />";
?>
```

retourne, en 2002,

```
Cette année, Pâques tombe le dimanche 31 mars
En 2001, Pâques était le dimanche 15 avril
En 1969, Pâques c'est le 6/4
```

Conversion d'une date d'un calendrier à l'autre

Toutes ces fonctions s'appuient sur une date exprimée dans le calendrier julien (et non le calendrier grégorien auquel nous sommes habitués).

Mais, pour l'instant, commençons par les fonctions de conversion de/vers le timestamp UNIX.

unixToJD()

Retourne le nombre de jours du calendrier julien à partir de la date donnée (ou, à défaut, la date courante) en timestamp UNIX.

Syntaxe	`int unixToJD([timestamp $date])`
`$date`	Date en nombre de secondes depuis le 1er janvier 1970.
`retour`	Nombre de jours du calendrier julien.

JDToUnix()

Retourne la date en timestamp UNIX à partir de la date donnée en nombre de jours du calendrier julien.

Syntaxe	`int JDToUnix(int $dateJulien)`
`$dateJulien`	Date en nombre de jours du calendrier julien.
`retour`	Date en timestamp UNIX.

Informations sur les différents calendriers
Vous trouverez plus d'informations sur ces différents calendriers à l'adresse : ***http://www.bdl.fr/Granpub/Calendriers.html****.*

Voici maintenant les autres fonctions de conversion de calendriers.

JDToGregorian()

Retourne une chaîne au format "mois/jour/année" du calendrier grégorien, correspondant au nombre de jours donnés dans le calendrier julien.

Syntaxe	`string JDToGregorian(int $jour)`
`$jour`	Nombre de jours dans le calendrier julien.
`retour`	Chaîne au format "mois/jour/année" du calendrier grégorien.

gregorianToJD()

Retourne le nombre de jours du calendrier julien correspondant à la date donnée dans le calendrier grégorien.

Syntaxe	`int gregorianToJD(int $mois, int $jour, int $annee)`
`$mois`	Mois dans le calendrier julien.
`$jour`	Jour dans le calendrier julien.
`$annee`	Année dans le calendrier julien.
`retour`	Nombre de jours du calendrier julien.

JDToJewish()

Retourne une chaîne au format "mois/jour/année" du calendrier juif correspondant au nombre de jours donnés dans le calendrier julien.

Syntaxe	`string JDToJewish(int $jour)`
`$jour`	Nombre de jours dans le calendrier julien.
`retour`	Chaîne au format "mois/jour/année" du calendrier juif.

jewishToJD()

Retourne le nombre de jours du calendrier julien correspondant à la date donnée dans le calendrier juif.

Syntaxe	`int jewishToJD(int $mois, int $jour, int $annee)`
`$mois`	Mois dans le calendrier juif.
`$jour`	Jour dans le calendrier juif.
`$annee`	Année dans le calendrier juif.
`retour`	Nombre de jours du calendrier julien.

JDToFrench()

Retourne une chaîne au format "mois/jour/année" du calendrier républicain français correspondant au nombre de jours donnés dans le calendrier julien.

Syntaxe	`string JDToFrench(int $jour)`
`$jour`	Nombre de jours dans le calendrier julien.
`retour`	Chaîne au format "mois/jour/année" du calendrier républicain français.

frenchToJD()

Retourne le nombre de jours du calendrier julien correspondant à la date donnée dans le calendrier républicain français (valable pour les dates du 22 septembre 1792 au 22 septembre 1806 du calendrier grégorien, même si son utilisation a été abandonnée le 31 décembre 1805).

Syntaxe	`int frenchToJD(int $mois, int $jour, int $annee)`
`$mois`	Mois dans le calendrier républicain français.
`$jour`	Jour dans le calendrier républicain français.
`$annee`	Année dans le calendrier républicain français.
`retour`	Nombre de jours du calendrier julien.

JDToJulian()

Retourne la date du calendrier julien au format "mois/jour/année" à partir de la date exprimée en nombre de jours.

Syntaxe	`string JDToJulian(int $jour)`
`$jour`	Nombre de jours dans le calendrier julien.
`retour`	Date du calendrier julien au format "mois/jour/année".

julianToJD()

Retourne la date du calendrier julien exprimée en nombre de jours.

Syntaxe	`int julianToJD(int $mois, int $jour, int $annee)`
`$mois`	Mois dans le calendrier julien.
`$jour`	Jour dans le calendrier julien.

`$annee`	Année dans le calendrier julien.
`retour`	Nombre de jours.

Manipulation d'une date dans le calendrier julien

JDMonthName()

Retourne en toutes lettres (en anglais) le nom du mois dans le calendrier précisé.

Syntaxe	`string JDMonthName(int $dateJulien, int $calendrier)`
`$dateJulien`	Date exprimée en nombre de jours du calendrier julien.
`$calendrier`	Une des valeurs suivantes : `0` = grégorien (abrégé aux trois premières lettres). `1` = grégorien. `2` = julien (abrégé aux trois premières lettres). `3` = julien. `4` = juif. `5` = républicain français (uniquement valable pour les dates allant du 22 septembre 1792 au 22 septembre 1806 du calendrier grégorien).
`retour`	Nom du mois.

JDDayOfWeek()

Retourne le nom (en anglais) du jour de la semaine ou son numéro.

Syntaxe	`mixed JDDayOfWeek(int $dateJulien, int $mode)`
`$dateJulien`	Date exprimée en nombre de jours du calendrier julien.
`$mode`	Une des valeurs suivantes : `0` = numéro du jour (1 = dimanche). `1` = nom du jour. `2` = nom du jour abrégé aux trois premières lettres.
`retour`	Numéro du jour de la semaine (mode = 0), nom du jour sinon.

8.3. Les gestionnaires d'événements

Jusqu'à l'arrivée de PHP 5, PHP proposait des bibliothèques permettant d'interagir avec des gestionnaires d'événements supportant le protocole ICAP (iCalendar Protocol).

Si vous utilisez PHP4, il vous est donc possible de créer votre propre interface Internet de suivi et d'administration de vos agendas. Cette interface vous sera alors accessible depuis n'importe quel poste disposant d'un navigateur (et non pas seulement depuis les postes équipés du logiciel client de la solution de gestionnaire d'événements retenue).

Pour cela, vous devrez utiliser la bibliothèque MCAL (Modular Calendar AccessLibrary). Celle-ci, en plus de supporter le protocole ICAP, permet d'utiliser des fichiers locaux comme support des données. De plus, cette bibliothèque a été développée pour être modulaire ce qui, en théorie du moins, lui permet de supporter d'autres protocoles.


INTERNET

En savoir plus
Le protocole ICAP est défini par les RFC-2445 "Internet Calendaring and Scheduling Core Object Specification (iCalendar)" et RFC-2446 "iCalendar Transport-Independent Interoperability Protocol (iTIP)" disponibles en anglais aux adresses suivantes :
http://www.ietf.org/rfc/rfc2445.txt
http://www.ietf.org/rfc/rfc2446.txt

Pour être utilisée, cette bibliothèque doit au préalable être récupérée, compilée et intégrée à PHP.

Installation

Cette bibliothèque n'est pas disponible sous Windows.

Sous Linux


REMARQUE

Pré-requis
Pour pouvoir compiler cette bibliothèque, il faut avoir installé au préalable les outils de compilation et autres commandes telles que "`flex`*".*

La bibliothèque `MCAL` est composée de deux éléments :

- `libmcal` : la bibliothèque proprement dite ;
- `mcaldrivers` : les modules de la bibliothèque comprenant les pilotes pour le protocole ICAP et ceux pour les fichiers locaux (`mstore`).

Ces deux fichiers (disponibles sur le CD-ROM) peuvent être téléchargés à l'adresse **http://sourceforge.net/projects/libmcal/**.

Après les avoir copiés dans un répertoire donné (ex. : */usr/local/src/lib*), décompressez ces deux fichiers à l'aide des commandes suivantes :

```
# gunzip libmcal-0.7.tar.gz
# tar xvf libmcal-0.7.tar
# gunzip mcaldrivers-0.9.tar.gz
# tar xvf mcaldrivers-0.9.tar
```

Puis, déplacez le contenu de `mcaldrivers` dans `libmcal` :

```
# mv mcal-drivers/* libmcal
```

Ensuite, passez à la génération des pilotes `mstore` et `icap` :

```
# cd libmcal
# cd mstore
# make
# cd ..
# cd icap
# make
# cd ..
```

Enfin, passez à la génération de la bibliothèque :

```
# chmod +x configure
# ./configure --with-mstore --with-icap
# make
```

Un fichier *libmcal.a* est alors généré.

Il est maintenant possible de (re)compiler PHP après avoir ajouté l'option "*--with-mcal=<chemin vers le répertoire libmcal contenant* `libmcal.a`" (ex. : `--with-mcal=/usr/local/src/lib/libmcal`).


RENVOI

Vous pouvez consulter le chapitre "Prise en main" pour plus de détails sur la compilation de PHP.


REMARQUE

Initialisation mstore

Si vous souhaitez utiliser le module `mstore` *(autrement dit utiliser un fichier local pour stocker vos agendas), il vous faut d'abord créer un répertoire pour héberger ces agendas, ainsi qu'un fichier de mots de passe. Pour cela, il suffit de suivre la procédure suivante :*

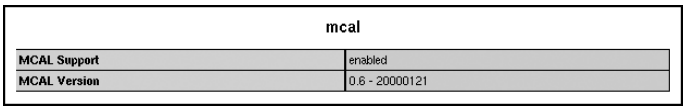
```
# mkdir /var/calendar
# chmod 1777 /var/calendar
# touch /etc/mpasswd
# chmod a+r /etc/mpasswd
```

`Mstore` *utilisant un fichier de mots de passe au format identique à celui d'Apache, vous devez utiliser la commande* `htpasswd` *fournie avec Apache pour le générer. Vous prendrez soin, ici, de remplacer la variable* `$APACHE_HOME` *par sa valeur (qui pourra être /usr/local/apache, /usr ou autre selon votre configuration).*

```
# $APACHE_HOME/bin/htpasswd -b /etc/mpasswd <login> <mot de passe>
```

Vérification

Vous pouvez vérifier que le support de la bibliothèque MCAL est effectif en appelant un script contenant simplement <?php phpinfo(); ?> et qui doit afficher :



mcal	
MCAL Support	enabled
MCAL Version	0.6 - 20000121

Figure 8.2 : *phpinfo()*

Les fonctions

Il est possible de distinguer deux types de fonctions : celles qui permettent de manipuler les dates et celles qui permettent effectivement de manipuler les informations stockées dans les agendas.

Les fonctions de manipulation de dates trouvent, pour la plupart, leur équivalent parmi les fonctions proposées par défaut avec PHP. Il y a toutefois une petite différence, puisque, contrairement aux autres, les fonctions de la bibliothèque MCAL sont également valables pour des dates précédant le 1er janvier 1970.

mcal_date_valid()

Teste la validité de la date.

Syntaxe	`boolean mcal_date_valid(int $annee, int $mois, int $jour)`
`$annee`	Année.
`$mois`	Mois.
`$jour`	Jour.
`retour`	`TRUE` si la date est valide, `FALSE` sinon.

Équivalent sans la bibliothèque MCAL :

```
checkdate($mois, $jour, $annee);
```

mcal_time_valid()

Teste la validité de l'heure.

Syntaxe	`boolean mcal_time_valid(int $heure, int $minute, int $seconde)`
`$heure`	Heure.
`$minute`	Minute.
`$seconde`	Seconde.

retour	TRUE si l'heure est valide, FALSE sinon.

Équivalent sans la bibliothèque MCAL :

```
(    ($heure>=0 && $heure<24 ) && ($minute>=0 && $minute<60)
  && ($seconde>=0 && $seconde<60))
```

mcal_day_of_week()

Retourne le numéro du jour dans la semaine de la date donnée.

Syntaxe	int mcal_day_of_week(int $annee, int $mois, int $jour)
$annee	Année.
$mois	Mois.
$jour	Jour.
retour	Numéro du jour dans la semaine (0 = dimanche, ..., 6 = samedi). En fait, des constantes existent MCAL_SUNDAY, MCAL_MONDAY, MCAL_TUESDAY, MCAL_THURSDAY, MCAL_FRIDAY, MCAL_Saturday.

Équivalent sans la bibliothèque MCAL (pour les dates dans la limite de validité des timestamps) :

```
$tab = getdate(mktime(0, 0, 0, $mois, $jour, $annee));
return $tab["wday"];
```

mcal_day_of_year()

Retourne le numéro du jour dans l'année de la date donnée.

Syntaxe	int mcal_day_of_year(int $annee, int $mois, int $jour)
$annee	Année.
$mois	Mois.
$jour	Jour.
retour	Numéro du jour dans l'année (1 = 1er janvier, ...).

Équivalent sans la bibliothèque MCAL (pour les dates dans la limite de validité des timestamps) :

```
$tab = getdate(mktime(0, 0, 0, $mois, $jour, $annee));
return $tab["yday"] + 1;
```

mcal_date_compare()

Compare deux dates.

Syntaxe	`int mcal_date_compare(int $annee1, int $mois1, int $jour1, int $annee2, int $mois2, int $jour2)`
`$annee1`	Année de la 1ère date à comparer.
`$mois1`	Mois de la 1ère date à comparer.
`$jour1`	Jour de la 1ère date à comparer.
`$annee2`	Année de la 2e date à comparer.
`$mois2`	Mois de la 2e date à comparer.
`$jour2`	Jour de la 2e date à comparer.
`retour`	`-1` si la 1ère date est antérieure à la seconde, `0` si elles sont identiques, `1` sinon.

Équivalent sans la bibliothèque `MCAL` (pour les dates dans la limite de validité des timestamps) :

```
$diff = mktime(0, 0, 0, $mois1, $jour1, $annee1) -
        mktime(0, 0, 0, $mois2, $jour2, $annee2);
if ($diff == 0) return 0;
return $diff>0?1:-1;
```

mcal_is_leap_year()

Teste si l'année est bissextile.

Syntaxe	`boolean mcal_is_leap_year(int $annee)`
`$annee`	Année.
`retour`	`TRUE` si l'année est bissextile, `FALSE` sinon.

Équivalent sans la bibliothèque `MCAL` :

```
return checkdate(2, 29, $annee);
```

mcal_days_in_month()

Retourne le nombre de jours dans le mois en tenant compte des années bissextiles.

Syntaxe	`int mcal_days_in_month(int $mois, boolean $bissextile)`
`$mois`	Mois.
`$bissextile`	Indiquez `TRUE` s'il s'agit d'une année bissextile, `FALSE` sinon.
`retour`	Nombre de jours dans le mois.

Équivalent sans la bibliothèque `MCAL` :

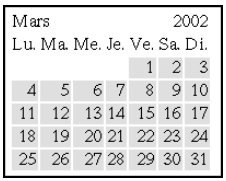
```
if ($mois == 2) {
  if ($bissextile) {
    return 29;
  } else {
    return 28;
  }
}
if (checkdate($mois, 31, 2000)) {
  return 31;
} else {
  return 30;
}
```

Une utilisation courante de ces fonctions consiste à afficher un calendrier (annuel, mensuel, etc.).

2002

Janv.	Févr.	Mars	Avril	Mai	Juin	Juil.	Août	Sept.	Oct.	Nov.	Déc.
Ma. 1	Ve. 1	Ve. 1	Lu. 1	Me. 1	Sa. 1	Lu. 1	Je. 1	Di. 1	Ma 1	Ve. 1	Di. 1
Me. 2	Sa. 2	Sa. 2	Ma. 2	Je. 2	Di. 2	Ma. 2	Ve. 2	Lu. 2	Me. 2	Sa. 2	Lu. 2
Je. 3	Di. 3	Di. 3	Me. 3	Ve. 3	Lu. 3	Me. 3	Sa. 3	Ma. 3	Je. 3	Di. 3	Ma. 3
Ve. 4	Lu. 4	Lu. 4	Je. 4	Sa. 4	Ma. 4	Je. 4	Di. 4	Me. 4	Ve. 4	Lu. 4	Me. 4
Sa. 5	Ma. 5	Ma. 5	Ve. 5	Di. 5	Me. 5	Ve. 5	Lu. 5	Je. 5	Sa. 5	Ma. 5	Je. 5
Di. 6	Me. 6	Me. 6	Sa. 6	Lu. 6	Je. 6	Sa. 6	Ma. 6	Ve. 6	Di. 6	Me. 6	Ve. 6
Lu. 7	Je. 7	Je. 7	Di. 7	Ma. 7	Ve. 7	Di. 7	Me. 7	Sa. 7	Lu. 7	Je. 7	Sa. 7
Ma. 8	Ve. 8	Ve. 8	Lu. 8	Me. 8	Sa. 8	Lu. 8	Je. 8	Di. 8	Ma. 8	Ve. 8	Di. 8
Me. 9	Sa. 9	Sa. 9	Ma. 9	Je. 9	Di. 9	Ma. 9	Ve. 9	Lu. 9	Me. 9	Sa. 9	Lu. 9
Je. 10	Di. 10	Di. 10	Me.10	Ve.10	Lu. 10	Me.10	Sa. 10	Ma.10	Je. 10	Di. 10	Ma.10
Ve.11	Lu. 11	Lu.11	Je. 11	Sa. 11	Ma.11	Je. 11	Di. 11	Me.11	Ve.11	Lu.11	Me.11
Sa. 12	Ma.12	Ma.12	Ve. 12	Di. 12	Me.12	Ve.12	Lu. 12	Je. 12	Sa. 12	Ma.12	Je. 12
Di. 13	Me.13	Me.13	Sa. 13	Lu. 13	Je. 13	Sa. 13	Ma.13	Ve.13	Di. 13	Me.13	Ve.13
Lu. 14	Je. 14	Je. 14	Di. 14	Ma.14	Ve. 14	Di. 14	Me.14	Sa. 14	Lu. 14	Je. 14	Sa. 14
Ma.15	Ve. 15	Ve.15	Lu. 15	Me.15	Sa. 15	Lu. 15	Je. 15	Di. 15	Ma.15	Ve.15	Di. 15
Me.16	Sa. 16	Sa. 16	Ma.16	Je. 16	Di. 16	Ma.16	Ve. 16	Lu. 16	Me.16	Sa. 16	Lu. 16
Je. 17	Di. 17	Di. 17	Me.17	Ve.17	Lu. 17	Me.17	Sa. 17	Ma.17	Je. 17	Di. 17	Ma.17
Ve.18	Lu. 18	Lu.18	Je. 18	Sa. 18	Ma.18	Je. 18	Di. 18	Me.18	Ve.18	Lu.18	Me.18
Sa. 19	Ma.19	Ma.19	Ve. 19	Di. 19	Me.19	Ve.19	Lu. 19	Je. 19	Sa. 19	Ma.19	Je. 19
Di. 20	Me.20	Me.20	Sa. 20	Lu. 20	Je. 20	Sa. 20	Ma.20	Ve. 20	Di. 20	Me.20	Ve. 20
Lu. 21	Je. 21	Je. 21	Di. 21	Ma.21	Ve. 21	Di. 21	Me.21	Sa. 21	Lu. 21	Je. 21	Sa. 21
Ma.22	Ve. 22	Ve.22	Lu. 22	Me.22	Sa. 22	Lu. 22	Je. 22	Di. 22	Ma.22	Ve.22	Di. 22
Me.23	Sa. 23	Sa. 23	Ma.23	Je. 23	Di. 23	Ma.23	Ve. 23	Lu. 23	Me.23	Sa. 23	Lu. 23
Je. 24	Di. 24	Di. 24	Me.24	Ve.24	Lu. 24	Me.24	Sa. 24	Ma.24	Je. 24	Di. 24	Ma.24
Ve.25	Lu. 25	Lu.25	Je. 25	Sa. 25	Ma.25	Je. 25	Di. 25	Me.25	Ve.25	Lu.25	Me.25
Sa. 26	Ma.26	Ma.26	Ve. 26	Di. 26	Me.26	Ve.26	Lu. 26	Je. 26	Sa. 26	Ma.26	Je. 26
Di. 27	Me.27	Me.27	Sa. 27	Lu. 27	Je. 27	Sa. 27	Ma.27	Ve.27	Di. 27	Me.27	Ve.27
Lu. 28	Je. 28	Je. 28	Di. 28	Ma.28	Ve. 28	Di. 28	Me.28	Sa. 28	Lu. 28	Je. 28	Sa. 28
Ma.29		Ve.29	Lu. 29	Me.29	Sa. 29	Lu. 29	Je. 29	Di. 29	Ma.29	Ve.29	Di. 29
Me.30		Sa. 30	Ma.30	Je. 30	Di. 30	Ma.30	Ve. 30	Lu. 30	Me.30	Sa. 30	Lu. 30
Je. 31		Di. 31		Ve.31		Me.31	Sa. 31		Je. 31		Ma.31

Figure 8.3 : *Calendrier annuel*



Mars						2002
Lu.	Ma.	Me.	Je.	Ve.	Sa.	Di.
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Figure 8.4 : *Calendrier mensuel*

Ce résultat peut être obtenu avec le script suivant :

Listing 8.9 : mcal_01_inc.php

```
<?php

class MCAL_Agenda {

    /**
     * Affiche une année sous forme de calendrier
     */
    function afficheAnnee($annee)
    {
        $labelMois = array(1 => "Janv.", "Févr.", "Mars", "Avril",
                           "Mai", "Juin", "Juil.", "Août",
                           "Sept.", "Oct.", "Nov.", "Déc.");
        $labelJour = array("Di.","Lu.","Ma.","Me.","Je.","Ve.","Sa.");

        echo "<table cellspacing=\"10\">\n";
        echo "<tr><td colspan=\"12\" align=\"center\">$annee</td></tr>\n";

        // Affiche les entêtes de colonnes (les mois)
        echo "<tr>";
        for ($mois = 1; $mois <= 12; $mois++) {
            echo "<td>".$labelMois[$mois]."</td>";
        }
        echo "</tr>\n";

        // Complète chaque mois
        echo "<tr>";
        for ($mois = 1; $mois <= 12; $mois++) {

            // Donne une couleur de fond différente pour les mois
            // pairs et les mois impairs
            if (($mois % 2) == 0) {
                $cssClass = "moisPair";
            } else {
                $cssClass = "moisImpair";
            }
            echo "<td class=\"$cssClass\">\n";
            echo "<table cellspacing=\"0\" cellpadding=\"0\">";

            // Détermine le nombre de jours dans le mois
            // en tenant compte des années bissextiles
```

```
        $nbJour = mcal_days_in_month($mois,mcal_is_leap_year($annee));
        for ($jour = 1; $jour <= $nbJour; $jour++) {
            echo "<tr>";
            echo "<td align=\"left\">";

            // Determine de quel jour il s'agit
            // pour l'afficher en toutes lettres
            echo $labelJour[mcal_day_of_week($annee,$mois,$jour)]."</td>";

            $cssClass = "date";
            echo "<td align=\"right\" class=\"$cssClass\">";
            echo "$jour</td>";
            echo "</tr>";
        }
        // Ajoute des blancs en bas de tableau si necessaire
        for ($jour = $nbJour + 1; $jour <= 31; $jour++) {
            echo "<tr><td> </td><td> </td></tr>";
        }
        echo "</table>\n";
        echo "</td>";
    }
    echo "</tr>";
    echo "</table>\n";
}

/**
 * Affiche un mois sous forme de calendrier
 */
function afficheMois($mois, $annee, $mcal="")
{
    $labelMois = array(1 => "Janvier", "Février", "Mars", "Avril",
                "Mai", "Juin", "Juillet", "Août", "Septembre",
                "Octobre", "Novembre", "Décembre");
    $labelJour = array("Lu.","Ma.","Me.","Je.","Ve.","Sa.","Di.");


    // Détermine le nombre de jours dans le mois
    // en tenant compte des années bissextiles
    $nbJours=mcal_days_in_month($mois, mcal_is_leap_year($annee));

    echo "<table>\n";
    echo "<tr><td colspan=\"4\">$labelMois[$mois]</td>";
    echo "<td colspan=\"3\" align=\"right\">$annee</td></tr>\n";
    echo "<tr>";
    for ($i = 0; $i < 7; $i++) {
        echo "<td>$labelJour[$i]</td>";
    }
    echo "</tr>\n";

    echo "<tr>";

    $premierJour = mcal_day_of_week($annee, $mois, 1);
```

```
        // convertir 0=Dimanche...6=Samedi
        // en 0=Lundi...6=Dimanche
        $premierJour = ($premierJour + 6) % 7;

        // Sauter autant de colonne que nécessaire pour atteindre
        // le premier jour de la semaine
        for ($i = 0; $i < $premierJour; $i++) {
            echo "<td></td>";
        }

        // Puis passer en revue tous les jours du mois
        for ($i = 0; $i < $nbJours; $i++) {
            if (($i + $premierJour) % 7 == 0) {
                // Retour à la ligne chaque Lundi
                echo "</tr>\n<tr>";
            }

            $cssClass = "date";
            echo "<td align=\"right\" class=\"$cssClass\">";
            echo ($i+1);
            echo "</td>";
        }

        echo "</tr>\n";
        echo "</table>\n";
    }
}
?>
```

Ce script pourra être appelé de la façon suivante (ici, pour afficher les deux types de calendrier) :

Listing 8.10 : mcal_01.php

```
<?php
    include("mcal_01_inc.php");
?>
<html>
<head>
<style>
.moisPair {
    background-color: #DDDDFF;
}
.moisImpair {
    background-color: #9999FF;
}
.date {
    background-color: #DDDDFF;
}
</style>
</head>
<body>
```

```
<h2>Bibliothèque MCAL</h2>
<?php
    $dateCourante = getdate();

    MCAL_Agenda::afficheAnnee($dateCourante["year"]);
    MCAL_Agenda::afficheMois($dateCourante["mon"], $dateCourante["year"]);
?>
</body>
</html>
```

Les fonctions d'accès aux agendas

Comme le veut la logique, pour accéder à un agenda, il faut d'abord se connecter. Il existe à ce propos trois fonctions : `mcal_open()`, `mcal_popen()`, et `mcal_reopen()`.

mcal_open()

Ouvre un calendrier.

Syntaxe	`resource mcal_open(string $calendrier, string $utilisateur, string $motDePasse[, int $options])`
`$calendrier`	Identifiant du calendrier que vous souhaitez ouvrir. Chaîne de la forme `{serveur/protocole}<proprietaire>calendrier`.
`$utilisateur`	Nom de l'utilisateur.
`$motDePasse`	Mot de passe de l'utilisateur.
`$options`	Manifestement inutilisé.
`retour`	Retourne un identifiant de connexion, ou `FALSE` en cas d'échec.

Cette fonction se décline également en `mcal_popen()` (mais la persistance de la connexion n'est pas démontrée) et `mcal_reopen()` (dont le fonctionnement ne semble pas tout à fait satisfaisant). Il n'est d'ailleurs pas assuré que leur développement soit tout à fait finalisé.

Dans le cas d'une connexion à un calendrier via le protocole `mstore`, le serveur est nécessairement local. La chaîne d'identification du serveur sera donc de la forme `{/mstore}<proprietaire>calendrier`.

Il est à noter qu'un utilisateur peut se connecter à l'agenda d'une autre personne pour accéder à ses informations publiques. D'où l'utilité possible du champ propriétaire. Toutefois, si ce paramètre n'est pas précisé, c'est le calendrier de l'utilisateur qui est considéré.

Nous ajouterons donc à notre classe `MCAL_Agenda` la méthode suivante :

Listing 8.11 : mcal_02_inc.php (extrait)

```
<?php

// Extrait de la Classe MCAL
```

```
class MCAL_Agenda {

    /**
     * Se connecte à l'agenda
     * et retourne l'identifiant associé
     */
    function connect() {
        $mcal = mcal_open("{/mstore}biblephp","biblephp","pwd")
        or die("La connexion à l'agenda a échoué");
        return $mcal;
    }

(...)

}
?>
```

Une fois toutes les opérations effectuées, il sera possible de libérer les ressources en fermant la connexion au calendrier par `mcal_close()`.

mcal_close()

Ferme la connexion au calendrier.

Syntaxe	`boolean mcal_close(resource $idConnexion [, int $options])`
`$idConnexion`	Identifiant de connexion tel que retourné par `mcal_open()`.
`$options`	Rôle indéterminé.
`retour`	Retourne `FALSE` en cas d'échec, `TRUE` sinon.

Lecture des événements

Une des premières opérations que vous serez amené à faire sera probablement de récupérer la liste des événements que vous avez notés dans l'agenda sur une période donnée. Pour cela, vous devrez faire appel à la fonction `mcal_list_events()`.

mcal_list_events()

Retourne la liste des événements intervenant entre deux dates.

Syntaxe	`array mcal_list_events(resource $idConnexion, int $anneeDebut, int $moisDebut, int $jourDebut, int $anneeFin, int $moisFin, int $jourFin)`

`$idConnexion`	Identifiant de connexion tel que retourné par `mcal_open()`.
`$anneeDebut`	Année de la date de début de recherche.
`$moisDebut`	Mois de la date de début de recherche.
`$jourDebut`	Jour de la date de début de recherche.
`$anneeFin`	Année de la date de fin de recherche.
`$moisFin`	Mois de la date de fin de recherche.
`$jourFin`	Jour de la date de fin de recherche.
`retour`	Tableau indexé ayant pour valeurs des entiers correspondant à des identifiants d'événements.

Cela nous permet déjà de créer une fonction (ou plus exactement une méthode) entraînant l'affichage d'un calendrier avec, en surbrillance, les jours auxquels des événements sont associés.

Voici donc la méthode d'affichage d'un calendrier enrichi par cette nouvelle fonctionnalité :

Listing 8.12 : mcal_02_inc.php (extrait)

```
<?php

// Extrait de la classe MCAL_Agenda

class MCAL_Agenda {

// (...)

    /**
     * Affiche une année sous forme de calendrier
     * et montre les dates associées à un événement
     */
    function afficheAnnee($annee, $mcal="")
    {
        $labelMois = array(1 => "Janv.", "Févr.", "Mars", "Avril",
                           "Mai", "Juin", "Juil.", "Août",
                           "Sept.", "Oct.", "Nov.", "Déc.");
        $labelJour = array("Di.","Lu.","Ma.","Me.","Je.","Ve.","Sa.");

        echo "<table cellspacing=\"10\">\n";
        echo "<tr><td colspan=\"12\" align=\"center\">$annee</td></tr>\n";

        // Affiche les entêtes de colonnes (les mois)
        echo "<tr>";
        for ($mois = 1; $mois <= 12; $mois++) {
            echo "<td>".$labelMois[$mois]."</td>";
        }
        echo "</tr>\n";

        // Complète chaque mois
        echo "<tr>";
```

```
for ($mois = 1; $mois <= 12; $mois++) {

    // Donne une couleur de fond différente pour les mois
    // pairs et les mois impairs
    if (($mois % 2) == 0) {
        $cssClass = "moisPair";
    } else {
        $cssClass = "moisImpair";
    }
    echo "<td class=\"$cssClass\">\n";
    echo "<table cellspacing=\"0\" cellpadding=\"0\">";

    // Détermine le nombre de jours dans le mois
    // en tenant compte des années bissextiles
    $nbJour = mcal_days_in_month($mois, mcal_is_leap_year($annee));
    for ($jour = 1; $jour <= $nbJour; $jour++) {
        echo "<tr>";
        echo "<td align=\"left\">";

        // Determine de quel jour il s'agit
        // pour l'afficher en toutes lettres
        echo $labelJour[mcal_day_of_week($annee,$mois,$jour)]."</td>";

        // A-t-on un événement lié à cette date ?
        $avecEvenement = FALSE;
        if ($mcal != "") {
            $evenements = mcal_list_events($mcal,
                                $annee, $mois, $jour,
                                $annee, $mois, $jour);
            if (count($evenements) >0 ) $avecEvenement = TRUE;
            ✂
            ✂
        }


        if ($avecEvenement) {
            $cssClass = "dateEvenement";
        } else {
            $cssClass = "date";
        }
        echo "<td align=\"right\" class=\"$cssClass\">";
        echo "$jour</td>";
        echo "</tr>";
    }
    // Ajoute des blancs en bas de tableau si nécessaire
    for ($jour = $nbJour + 1; $jour <= 31; $jour++) {
        echo "<tr><td> </td><td> </td></tr>";
    }
    echo "</table>\n";
    echo "</td>";
}
echo "</tr>";
```

```
        echo "</table>\n";
    }

// (...)
}
?>
```

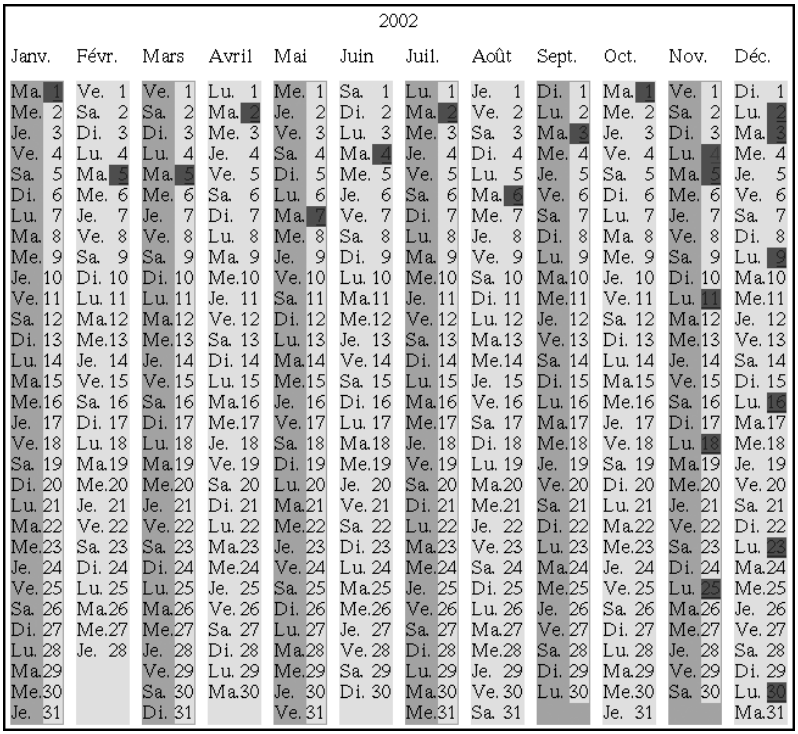
qui pourra être appelé par :

Listing 8.13 : mcal_02.php

```
<?php
include("mcal_02_inc.php");
?>

<html>
<head>
<style>
.moisPair {
    background-color: #DDDDFF;
}
.moisImpair {
    background-color: #9999FF;
}
.date {
    background-color: #DDDDFF;
}
.dateEvenement {
    background-color: #FF0000;
}
</style>
</head>
<body>
<h2>Bibliothèque MCAL</h2>
<?php

    $mcal          = MCAL_Agenda::connect();

    $dateCourante = getdate();

    MCAL_Agenda::afficheAnnee($dateCourante["year"],
                              $mcal);
    MCAL_Agenda::afficheMois($dateCourante["mon"],
                             $dateCourante["year"],
                             $mcal);
?>
</body>
</html>
```

Le fichier *mcal_02.php* n'est rien d'autre que *mcal_01.php* auquel nous avons ajouté un style afin de préciser la couleur des dates associées à un événement et une connexion à l'agenda.

Voici un exemple de résultat obtenu (dans les cas d'un événement mensuel sur toute l'année et d'un événement hebdomadaire sur les derniers mois).



2002

Janv.	Févr.	Mars	Avril	Mai	Juin	Juil.	Août	Sept.	Oct.	Nov.	Déc.
Ma. 1	Ve. 1	Ve. 1	Lu. 1	Me. 1	Sa. 1	Lu. 1	Je. 1	Di. 1	Ma. 1	Ve. 1	Di. 1
Me. 2	Sa. 2	Sa. 2	Ma. 2	Je. 2	Di. 2	Ma. 2	Ve. 2	Lu. 2	Me. 2	Sa. 2	Lu. 2
Je. 3	Di. 3	Di. 3	Me. 3	Ve. 3	Lu. 3	Me. 3	Sa. 3	Ma. 3	Je. 3	Di. 3	Ma. 3
Ve. 4	Lu. 4	Lu. 4	Je. 4	Sa. 4	Ma. 4	Je. 4	Di. 4	Me. 4	Ve. 4	Lu.	Me. 4
Sa. 5	Ma. 5	Ma. 5	Ve. 5	Di. 5	Me. 5	Ve. 5	Lu. 5	Je. 5	Sa. 5	Ma. 5	Je. 5
Di. 6	Me. 6	Me. 6	Sa. 6	Lu. 6	Je. 6	Sa. 6	Ma. 6	Ve. 6	Di. 6	Me. 6	Ve. 6
Lu. 7	Je. 7	Je. 7	Di. 7	Ma. 7	Ve. 7	Di. 7	Me. 7	Sa. 7	Lu. 7	Je. 7	Sa. 7
Ma. 8	Ve. 8	Ve. 8	Lu. 8	Me. 8	Sa. 8	Lu. 8	Je. 8	Di. 8	Ma. 8	Ve. 8	Di. 8
Me. 9	Sa. 9	Sa. 9	Ma. 9	Je. 9	Di. 9	Ma. 9	Ve. 9	Lu. 9	Me. 9	Sa. 9	Lu. 9
Je. 10	Di. 10	Di. 10	Me.10	Ve.10	Lu. 10	Me.10	Sa. 10	Ma.10	Je. 10	Di. 10	Ma.10
Ve. 11	Lu. 11	Lu. 11	Je. 11	Sa. 11	Ma.11	Je. 11	Di. 11	Me.11	Ve. 11	Lu. 11	Me.11
Sa. 12	Ma.12	Ma.12	Ve. 12	Di. 12	Me.12	Ve. 12	Lu. 12	Je. 12	Sa. 12	Ma.12	Je. 12
Di. 13	Me.13	Me.13	Sa. 13	Lu. 13	Je. 13	Sa. 13	Ma.13	Ve. 13	Di. 13	Me.13	Ve. 13
Lu. 14	Je. 14	Je. 14	Di. 14	Ma.14	Ve. 14	Di. 14	Me.14	Sa. 14	Lu. 14	Je. 14	Sa. 14
Ma.15	Ve. 15	Ve. 15	Lu. 15	Me.15	Sa. 15	Lu. 15	Je. 15	Di. 15	Ma.15	Ve. 15	Di. 15
Me.16	Sa. 16	Sa. 16	Ma.16	Je. 16	Di. 16	Ma.16	Ve. 16	Lu. 16	Me.16	Sa. 16	Lu. 16
Je. 17	Di. 17	Di. 17	Me.17	Ve.17	Lu. 17	Me.17	Sa. 17	Ma.17	Je. 17	Di. 17	Ma.17
Ve. 18	Lu. 18	Lu. 18	Je. 18	Sa. 18	Ma.18	Je. 18	Di. 18	Me.18	Ve. 18	Lu. 18	Me.18
Sa. 19	Ma.19	Ma.19	Ve. 19	Di. 19	Me.19	Ve. 19	Lu. 19	Je. 19	Sa. 19	Ma.19	Je. 19
Di. 20	Me.20	Me.20	Sa. 20	Lu. 20	Je. 20	Sa. 20	Ma.20	Ve. 20	Di. 20	Me.20	Ve. 20
Lu. 21	Je. 21	Je. 21	Di. 21	Ma.21	Ve. 21	Di. 21	Me.21	Sa. 21	Lu. 21	Je. 21	Sa. 21
Ma.22	Ve. 22	Ve. 22	Lu. 22	Me.22	Sa. 22	Lu. 22	Je. 22	Di. 22	Ma.22	Ve. 22	Di. 22
Me.23	Sa. 23	Sa. 23	Ma.23	Je. 23	Di. 23	Ma.23	Ve. 23	Lu. 23	Me.23	Sa. 23	Lu. 23
Je. 24	Di. 24	Di. 24	Me.24	Ve.24	Lu. 24	Me.24	Sa. 24	Ma.24	Je. 24	Di. 24	Ma.24
Ve. 25	Lu. 25	Lu. 25	Je. 25	Sa. 25	Ma.25	Je. 25	Di. 25	Me.25	Ve. 25	Lu. 25	Me.25
Sa. 26	Ma.26	Ma.26	Ve. 26	Di. 26	Me.26	Ve. 26	Lu. 26	Je. 26	Sa. 26	Ma.26	Je. 26
Di. 27	Me.27	Me.27	Sa. 27	Lu. 27	Je. 27	Sa. 27	Ma.27	Ve. 27	Di. 27	Me.27	Ve. 27
Lu. 28	Je. 28	Je. 28	Di. 28	Ma.28	Ve. 28	Di. 28	Me.28	Sa. 28	Lu. 28	Je. 28	Sa. 28
Ma.29		Ve. 29	Lu. 29	Me.29	Sa. 29	Lu. 29	Je. 29	Di. 29	Ma.29	Ve. 29	Di. 29
Me.30		Sa. 30	Ma.30	Je. 30	Di. 30	Ma.30	Ve. 30	Lu. 30	Me.30	Sa. 30	Lu. 30
Je. 31		Di. 31		Ve. 31		Me.31	Sa. 31		Je. 31		Ma.31

Figure 8.5 : *Calendrier annuel avec des événements*

REMARQUE

Attention aux événements récurrents

Nous aurions pu (comme nous l'avons vu dans des exemples proposés sur Internet) éviter un appel à la fonction `mcal_list_event()` *pour chaque jour, et se contenter de l'appliquer à l'ensemble de la période considérée. Le problème est que, dans ce cas, pour les événements récurrents, nous n'aurions pu calculer leurs différentes dates d'apparition (la fonction* `mcal_list_event()` *ne retournant qu'une seule fois l'identifiant de l'événement, quel que soit son nombre d'apparitions sur la période donnée). Cette méthode n'est donc sérieusement envisageable que dans le cas d'un agenda ne contenant que des événements ponctuels.*

Comme vous le constatez, la fonction `mcal_list_event()` ne retourne qu'une liste d'identifiants, mais pas le descriptif des événements. Pour cela, vous devez faire appel (pour chaque identifiant) à la fonction `mcal_fetch_event()`.

mcal_fetch_event()

Retourne les propriétés d'un événement.

Syntaxe	`object mcal_fetch_event(resource $idConnexion, int $idEvenement [, int $options])`
`$idConnexion`	Identifiant de connexion tel que retourné par `mcal_open()`.
`$idEvenement`	Identifiant de l'événement.
`$options`	Manifestement inutilisé.
`retour`	Objet *événement*.

Les propriétés d'un événement sont donc retournées sous la forme d'un objet.

Cet objet possède les attributs publics suivants:

- `id` (int) : identifiant de l'événement ;
- `public` (boolean) : `TRUE` si l'événement est public, `FALSE` s'il est privé ;
- `title` (string) : nom de l'événement ;
- `category` (string) : chaîne de caractères libre destinée à indiquer la catégorie de l'événement ;
- `description` (string) : description de l'événement ;
- `attrlist` (array) : tableau associatif des attributs personnalisés (attribut => valeur) ;
- `start` (objet *date et heure*) : date de début (de la première occurrence) de l'événement ;
- `end` (objet *date et heure*) : date de fin (de la première occurrence) de l'événement ;
- `recur_type` (int) : type de fréquence pouvant avoir une des valeurs `MCAL_RECUR_NONE` (ponctuelle), `MCAL_RECUR_DAILY` (fréquence exprimée en jours), `MCAL_RECUR_WEEKLY` (fréquence exprimée en semaines), `MCAL_RECUR_MONTHLY_MDAY` (fréquence exprimée en mois), `MCAL_RECUR_MONTHLY_WDAY` (fréquence exprimée en mois, basée sur le jour de la semaine), `MCAL_RECUR_YEARLY` (fréquence exprimée en années).
- `recur_interval` (int) : fréquence 1 (jour/semaine/mois/année) sur `recur_interval` (ex. : un jour sur deux si `recur_type = MCAL_RECUR_DAILY` et `recur_interval = 2`) ;
- `recur_data` (int) : jours de la semaine pendant lesquels l'événement a lieu (si `recur_type = MCAL_RECUR_WEEKLY`). Cette valeur est la somme des valeurs suivantes : `1` = dimanche, `2` = lundi, `4` = mardi, `8` = mercredi, `16` = jeudi, `32` = vendredi, `64` = samedi).
- `recur_enddate` (objet *date et heure*) : date à laquelle l'événement récurrent s'arrête ;
- `alarm` (int) : délai (en minutes) précédant l'événement pour lequel il faut déclencher une alarme.

L'objet *date et heure*, quant à lui, possède les attributs publics suivants :

- `year` (int) : année ;
- `month` (int) : mois ;

- `mday` (int) : jour ;
- `hour` (int) : heure ;
- `min` (int) : minutes ;
- `sec` (min) : secondes ;
- `alarm` (min) : ce paramètre n'est toutefois pas utilisé.

Nous pouvons donc ajouter une méthode affichant (brut de fonderie) les propriétés d'un événement.

Listing 8.14 : mcal_03_inc.php (extrait)

```
<?php

// Extrait de la Classe MCAL_Agenda

class MCAL_Agenda {

// (...)

    function afficheEvenement($idEvenement, $mcal)
    {
        $evenement = mcal_fetch_event($mcal, $idEvenement);
        echo "<table>";
        echo "<td colspan=\"2\" class=\"titreEvenement\">";
        echo $evenement->title." (".$evenement->id."/";
        if ($evenement->public) {
            echo "Publique";
        } else {
            echo "Privé";
        }
        echo ")</td></tr>\n";
        echo "<tr><td>Catégorie</td><td>".$evenement->category."</td></tr>";
        echo "<tr><td>Description</td>";
        echo "<td>".$evenement->description."</td></tr>";
        echo "<tr><td>Attributs Personnalisés</td>";
        echo "<td>";
        foreach ($evenement->attrlist as $nom => $valeur) {
            echo "$nom = $valeur<br />";
        }
        echo "</td></tr>";
        echo "<tr><td>Alarme</td>";
        echo "<td>".$evenement->alarm." minutes avant</td></tr>";

        $dateDebut = $evenement->start;
        echo "<tr><td>Début</td>";
        echo "<td>".$dateDebut->mday."/".$dateDebut->month."/";
        echo $dateDebut->year." ".$dateDebut->hour.":".$dateDebut->min;
        echo ":".$dateDebut->sec.;
        echo " (Alarme : ".$dateDebut->alarm.")</td></tr>";
```

```
$dateFin = $evenement->end;
echo "<tr><td>Fin</td><td>".$dateFin->mday."/".$dateFin->month."/";
echo $dateFin->year." ".$dateFin->hour.":".$dateFin->min;
echo ":".$dateFin->sec." (Alarme : ".$dateFin->alarm.")</td></tr>";
echo "<tr><td>Fréquence</td>";
echo "<td>";
switch ($evenement->recur_type) {
    case 0 : echo "Ponctuel";
             break;
    case 1 :
             if ($evenement->recur_interval == 1) {
                echo "Quotidien";
             } else {
                echo "1 jour sur ".$evenement->recur_interval;
             }
             break;
    case 2 : if ($evenement->recur_interval == 1) {
                echo "Hebdomadaire";
             } else {
                echo "1 semaine sur ".$evenement->recur_interval;
             }
             echo "<br />Les ";
             if (($evenement->recur_data &  1)>0) echo "Dimanche ";
             if (($evenement->recur_data &  2)>0) echo "Lundi ";
             if (($evenement->recur_data &  4)>0) echo "Mardi ";
             if (($evenement->recur_data &  8)>0) echo "Mercredi ";
             if (($evenement->recur_data & 16)>0) echo "Jeudi ";
             if (($evenement->recur_data & 32)>0) echo "Vendredi ";
             if (($evenement->recur_data & 64)>0) echo "Samedi ";
             break;
    case 3 : if ($evenement->recur_interval == 1) {
                echo "Mensuel";
             } else {
                echo "1 mois sur ".$evenement->recur_interval;
             }
             break;
    case 4 : echo "Tous les N-ième jour (Dim, Lun...) du mois.";
             if ($evenement->recur_interval == 1) {
                echo "Mensuel";
             } else {
                echo "1 mois sur ".$evenement->recur_interval;
             }
             break;
    case 5 : if ($evenement->recur_interval == 1) {
                echo "Annuel";
             } else {
                echo "1 an sur ".$evenement->recur_interval;
             }
             break;
    default : echo "????";
             break;
```

```
        }
        echo "</td></tr>";

        $dateFinRecur = $evenement->recur_enddate;
        echo "<tr><td>Jusqu'au</td>";
        echo "<td>".$dateFinRecur->mday."/".$dateFinRecur->month."/";
        echo $dateFinRecur->year." ";
        echo $dateFinRecur->hour.":".$dateFinRecur->min;
        echo ":".$dateFinRecur->sec;
        echo " (Alarme : ".$dateFinRecur->alarm.")</td></tr>";
        echo "</table>";
    }
}
?>
```

Bibliothèque MCAL

Réunion (1016137658/Privé)	
Catégorie	boulot
Description	Réunion d'avancement sur le projet X245
Attributs Personnalisés	monAttribut1 = A204
Alarme	5 minutes avant
Début	4/11/2002 10:0:0 (Alarme :)
Fin	4/11/2002 12:30:0 (Alarme :)
Fréquence	Hebdomadaire Les Lundi
Jusqu'au	31/12/2002 :: (Alarme :)

Figure 8.6 :
Fiche événement

Il est également possible de connaître la date du prochain événement par rapport à une date donnée.

mcal_next_recurrence()

Retourne la date de l'événement suivant la date donnée.

Syntaxe	`object mcal_next_recurrence(resource $idConnexion, int $weekstart, array $dateDebut)`
`$idConnexion`	Identifiant de connexion tel que retourné par `mcal_open()`.
`$weekstart`	Vraiment pas clair, le rôle de cet argument ! Il ne semble avoir aucune influence sur le résultat…
`$dateDebut`	Tableau associatif contenant la date de début de recherche avec les clés : "`year`" pour l'année. "`month`" pour le mois. "`mday`" pour le jour. "`hour`" pour l'heure. "`min`" pour les minutes. "`sec`" pour les secondes.
`retour`	Objet *date.*

Définition des données des événements

L'insertion de nouveaux événements dans l'agenda débute par l'initialisation des données de l'événement en cours de description par un appel à `mcal_event_init()`.

mcal_event_init()

Initialise une nouvelle description d'événement.

Syntaxe	`boolean mcal_event_init(resource $idConnexion)`
`$idConnexion`	Identifiant de connexion tel que retourné par `mcal_open()`.
`retour`	`TRUE`.

Il est ensuite possible de définir, pour l'événement en cours :

- Sa visibilité (publique ou privée) ;
- Son titre ;
- Sa description ;
- Sa catégorie ;
- Ses paramètres personnalisés ;
- Sa date.

mcal_event_set_class()

Fixe la visibilité (publique ou privée) de l'événement. Par défaut, un événement est privé.

Syntaxe	`boolean mcal_event_set_class(resource $idConnexion, boolean $visibilite)`
`$idConnexion`	Identifiant de connexion tel que retourné par `mcal_open()`.
`$visibilite`	Visibilité de l'événement (`TRUE` = publique, `FALSE` = privée).
`retour`	`TRUE`.

mcal_event_set_title()

Associe un nom à l'événement.

Syntaxe	`boolean mcal_event_set_title(resource $idConnexion, string $nom)`

`$idConnexion`	Identifiant de connexion tel que retourné par `mcal_open()`.
`$nom`	Intitulé de l'événement.
`retour`	`TRUE`.

mcal_event_set_description()

Associe un commentaire à l'événement.

Syntaxe	`boolean mcal_event_set_description(resource $idConnexion, string $description)`
`$idConnexion`	Identifiant de connexion tel que retourné par `mcal_open()`.
`$description`	Descriptif de l'événement.
`retour`	`TRUE`.

mcal_event_set_category()

Associe une catégorie à l'événement.

Syntaxe	`boolean mcal_event_set_category(resource $idConnexion, string $category)`
`$idConnexion`	Identifiant de connexion tel que retourné par `mcal_open()`.
`$category`	Catégorie de l'événement (il n'y a pas de valeurs prédéfinies : à vous de les créer).
`retour`	`TRUE`.

mcal_event_add_attribute()

Associe à l'événement un attribut personnalisé.

Syntaxe	`boolean mcal_event_add_attribute(resource $mcal, string $attribut, string $valeur)`
`$idConnexion`	Identifiant de connexion tel que retourné par `mcal_open()`.
`$attribut`	Nom de l'attribut personnalisé.
`$valeur`	Valeur associée à l'attribut personnalisé.

Définition des dates d'apparition des événements

Les dates des événements s'appuient sur plusieurs éléments. En effet, les événements n'étant pas toujours ponctuels, préciser des dates de début et de fin se révèle souvent insuffisant. Vous serez donc probablement amené à devoir préciser une fréquence d'apparition de l'événement. Un événement pourra, en effet, avoir lieu un jour sur N, une semaine sur N, un mois sur N (en s'appuyant soit sur le numéro du jour dans le mois, soit sur le nom du jour de la semaine et le numéro de la semaine dans le mois) ou bien encore un an sur N.

mcal_event_set_start()

Précise la date de début (de la première occurrence) de l'événement.

Syntaxe	`boolean mcal_event_set_start(resource $idConnexion, int $annee, int $mois [, int $jour [, int $heure [, int $minute [, int $seconde]]]])`
`$idConnexion`	Identifiant de connexion tel que retourné par `mcal_open()`.
`$annee`	Année de début de l'événement.
`$mois`	Mois de début de l'événement.
`$jour`	Jour de début de l'événement.
`$heure`	Heure de début de l'événement.
`$minute`	Minutes de la date de début de l'événement.
`$seconde`	Secondes de la date de début de l'événement.
`retour`	`TRUE`.

mcal_event_set_end()

Précise la date de fin (de la première occurrence) de l'événement.

Syntaxe	`boolean mcal_event_set_end(resource $idConnexion, int $annee, int $mois [, int $jour [, int $heure [, int $minute [, int $seconde]]]])`
`$idConnexion`	Identifiant de connexion tel que retourné par `mcal_open()`.
`$annee`	Année de début de l'événement.
`$mois`	Mois de début de l'événement.
`$jour`	Jour de début de l'événement.
`$heure`	Heure de début de l'événement.
`$minute`	Minutes de la date de fin de l'événement.
`$seconde`	Secondes de la date de fin de l'événement.
`retour`	`TRUE`.

mcal_event_set_recur_daily()

Précise la fréquence de l'événement sur la base des jours.

Syntaxe	`boolean mcal_event_set_recur_daily(resource $idConnexion, int $annee, int $mois, int $jour, int $frequence)`
`$idConnexion`	Identifiant de connexion tel que retourné par `mcal_open()`.
`$annee`	Année de la date de fin de récurrence de l'événement.
`$mois`	Mois de la date de fin de récurrence de l'événement.
`$jour`	Jour de la date de fin de récurrence de l'événement.
`$frequence`	L'événement a lieu un jour sur `$frequence`. (Si `$frequence = 0`, l'événement est ponctuel.)
`retour`	`TRUE`.

mcal_event_set_recur_weekly()

Précise la fréquence de l'événement sur la base des semaines.

Syntaxe	`boolean mcal_event_set_recur_weekly(resource $idConnexion, int $annee, int $mois, int $jour, int $frequence, int $joursSemaine)`
`$idConnexion`	Identifiant de connexion tel que retourné par `mcal_open()`.
`$annee`	Année de la date de fin de récurrence de l'événement.
`$mois`	Mois de la date de fin de récurrence de l'événement.
`$jour`	Jour de la date de fin de récurrence de l'événement.
`$frequence`	L'événement a lieu une semaine sur `$frequence`.
`$joursSemaine`	Indique quels jours de la semaine l'événement a lieu. Cette valeur est l'addition des valeurs suivantes : `1` = dimanche, `2` = lundi, `4` = mardi, `8` = mercredi, `16` = jeudi, `32` = vendredi, `64` = samedi.
`retour`	`TRUE`.

mcal_event_set_recur_monthly_mday()

Précise la fréquence de l'événement sur la base des mois.

Syntaxe	`boolean mcal_event_set_recur_monthly_mday(resource $idConnexion, int $annee, int $mois, int $jour, int $frequence)`

`$idConnexion`	Identifiant de connexion tel que retourné par `mcal_open()`.
`$annee`	Année de la date de fin de récurrence de l'événement.
`$mois`	Mois de la date de fin de récurrence de l'événement.
`$jour`	Jour de la date de fin de récurrence de l'événement.
`$frequence`	L'événement a lieu un mois sur `$frequence` (à la même date que le jour de début de la première occurrence).
`retour`	`TRUE`.

Semaines incomplètes

Le comportement de cette fonction peut surprendre dans certains cas (même s'il reste assez logique). Si vous prenez l'année 2003, qui commence par un mercredi, et que vous souhaitez indiquer que, tout au long de l'année, vous aurez une réunion le premier lundi du mois, vous prenez comme date de début de l'événement le lundi 6 janvier (premier lundi du mois de janvier, mais lundi de la seconde semaine du mois) : vous pointerez donc chaque mois sur le lundi de la seconde semaine du mois.

mcal_event_set_recur_monthly_wday()

Précise la fréquence de l'événement sur la base des mois (au même jour de la même semaine chaque mois).

Syntaxe	`boolean mcal_event_set_recur_monthly_wday(resource $idConnexion, int $annee, int $mois, int $jour, int $frequence)`
`$idConnexion`	Identifiant de connexion tel que retourné par `mcal_open()`.
`$annee`	Année de la date de fin de récurrence de l'événement.
`$mois`	Mois de la date de fin de récurrence de l'événement.
`$jour`	Jour de la date de fin de récurrence de l'événement.
`$frequence`	L'événement a lieu un mois sur `$frequence` (le même jour de la même semaine que la date de début de la première occurrence).
`retour`	`TRUE`.

mcal_event_set_recur_yearly()

Précise la fréquence de l'événement sur la base des années.

Syntaxe	`boolean mcal_event_set_recur_yearly(resource $idConnexion, int $annee, int $mois, int $jour, int $frequence)`

`$idConnexion`	Identifiant de connexion tel que retourné par `mcal_open()`.
`$annee`	Année de la date de fin de récurrence de l'événement.
`$mois`	Mois de la date de fin de récurrence de l'événement.
`$jour`	Jour de la date de fin de récurrence de l'événement.
`$frequence`	L'événement a lieu un an sur `$frequence`.
`retour`	`TRUE`.

Déclenchement de l'alarme

mcal_event_set_alarm()

Prépare une alarme pour l'événement.

Syntaxe	`boolean mcal_event_set_alarm(resource $idConnexion, int $delaisRappel)`
`$idConnexion`	Identifiant de connexion tel que retourné par `mcal_open()`.
`$delaisRappel`	Indique, en minutes, combien de temps avant l'échéance l'alarme doit se déclencher.
`retour`	`TRUE`.

Synthèse des propriétés d'un événement

Nous vous proposons de regrouper toutes ces propriétés dans un objet "fait maison". L'intérêt en est peut-être limité, mais, comme il s'agit d'un objet, vous pouvez toujours l'enrichir à votre guise. Il permet toutefois de définir les propriétés par des noms d'attributs en français (pour les non anglophones) et, surtout, de distinguer la définition de l'objet (assignation des valeurs des attributs) des "appels" aux calendriers (avec passage du paramètre `$idConnexion`). Cette dernière opération est ici réalisée par une unique méthode `prepare($idConnexion)`.

Listing 8.15 : mcal_evenement_inc.php (extrait)

```
<?php

// Extrait de MCAL_Evenement

class MCAL_Evenement {

    var $visibilite;
    var $titre, $description, $categorie;
    var $dateDebut, $heureDebut;
    var $dateFin, $heureFin;
    var $frequenceType;
    var $frequence;
```

```
var $frequenceDateFin;
var $frequenceJours;
var $alarme;
// Les attributs personnalisés
var $monAttribut1, $monAttribut2;

// Libre à vous de construire
// les méthodes setXX() et getXX() qui vont bien


// Stocke tous les attributs de l'objet
// dans l'événement courant
// (avant sauvegarde)
function prepare($mcal)
{
    mcal_event_init($mcal);

    if (isset($this->visibilite))
        mcal_event_set_class($mcal,
                             $this->visibilite);

    if (isset($this->titre))
        mcal_event_set_title($mcal, $this->titre);

    if (isset($this->description))
        mcal_event_set_description($mcal,
                             $this->description);

    if (isset($this->categorie)) {
        mcal_event_set_category($mcal,
                             $this->categorie);
    }

    if (isset($this->dateDebut)) {
        list($annee, $mois, $jour) =
            explode("-", $this->dateDebut);
        if (isset($this->heureDebut)) {
            list($heure, $minutes, $secondes) =
                explode(":", $this->heureDebut);
        }
        mcal_event_set_start($mcal,
            $annee, $mois, $jour, $heure, $minutes, $secondes);
    }

    if (isset($this->dateFin)) {
        list($annee, $mois, $jour) =
            explode("-", $this->dateFin);
        if (isset($this->heureFin)) {
            list($heure, $minutes, $secondes) =
                explode(":", $this->heureFin);
        }
        mcal_event_set_end($mcal,
```

```
            $annee, $mois, $jour, $heure, $minutes, $secondes);
    }

    if (isset($this->frequenceType) &&
        isset($this->frequenceDateFin) &&
        isset($this->frequence)) {

        if (isset($this->frequenceDateFin)) {
            list($annee, $mois, $jour) =
                explode("-", $this->frequenceDateFin);
        }

        switch ($this->frequenceType) {
            case MCAL_RECUR_DAILY :
                mcal_event_set_recur_daily($mcal,
                    $annee, $mois, $jour,
                    $this->frequence);
                break;
            case MCAL_RECUR_WEEKLY :
                mcal_event_set_recur_weekly($mcal,
                    $annee, $mois, $jour,
                    $this->frequence,
                    $this->frequenceJours);
                break;
            case MCAL_RECUR_MONTHLY_MDAY :
                mcal_event_set_recur_monthly_mday($mcal,
                    $annee, $mois, $jour,
                    $this->frequence);
                break;
            case MCAL_RECUR_MONTHLY_WDAY :
                mcal_event_set_recur_monthly_wday($mcal,
                    $annee, $mois, $jour,
                    $this->frequence);
                break;
            case MCAL_RECUR_YEARLY :
                mcal_event_set_recur_yearly($mcal,
                    $annee, $mois, $jour,
                    $this->frequence);
                break;
            default:
                // événement ponctuel
                break;
        }
    }
    if (isset($this->alarme)) {
       mcal_event_set_alarm($mcal, $this->alarme);
    }

    if (isset($this->monAttribut1)) {
        mcal_event_add_attribute($mcal,
            "monAttribut1",$this->monAttribut1);
    }
```

```
        if (isset($this->monAttribut2)) {
            mcal_event_add_attribute($mcal,
                "monAttribut2",$this->monAttribut2);
        }
    }
// (...)
}
```

À noter que, dans notre cas, nous avons un objet qui pourra contenir deux attributs personnalisés, que nous appellerons respectivement `monAttribut1` et `monAttribut2`.

Nous verrons un exemple d'utilisation de cet objet un peu plus loin dans ce chapitre.

Consultation de l'événement en cours

Il est possible à tout moment de consulter le contenu de l'événement en cours grâce à `mcal_fetch_current_stream_event()`. Ceci peut être utile pour le débogage, afin de vérifier ce que nous nous apprêtons à sauvegarder.

mcal_fetch_current_stream_event()

Retourne les propriétés de l'objet en cours.

Syntaxe	`object mcal_fetch_current_stream_event(resource $idConnexion)`
`$idConnexion`	Identifiant de connexion tel que retourné par `mcal_open()`.
`retour`	Objet *événement*.

À cette fin, nous avons ajouté une méthode à l'objet *MCAL_Evenement* évoqué précédemment.

Listing 8.16 : mcal_evenement_inc.php (extrait)

```
<?php

// Extrait de MCAL_Evenement

class MCAL_Evenement {

// (...)

    function toHTMLStringCurrentEvent($mcal) {
        $even = mcal_fetch_current_stream_event($mcal);

        $txt  = "";
        $txt .= "Id = ".$even->id."<br />";
        $txt .= "Titre = ".$even->title."<br />";
        $txt .= "Visibilité = ".$even->public."<br />";
        $txt .= "Categorie = ".$even->category."<br />";
```

```
        $txt .= "Description = ".$even->description."<br />";

        $attrlist = $even->attrlist;
        $txt .= "monAttribut1 = ".$attrlist["monAttribut1"]."<br />";
        $txt .= "monAttribut2 = ".$attrlist["monAttribut2"]."<br />";

        $date = $even->start;
        $txt .= "debut = ";
        $txt .= $date->year."-".$date->month."-".$date->mday;
        $txt .= " ".$date->hour.":".$date->min.":".$date->sec;
        $txt .= "<br />";

        $date = $even->end;
        $txt .= "debut = ";
        $txt .= $date->year."-".$date->month."-".$date->mday;
        $txt .= " ".$date->hour.":".$date->min.":".$date->sec;
        $txt .= "<br />";

        $txt .= "frequenceType = ".$even->recur_type."<br />";
        $txt .= "frequence = ".$even->recur_interval."<br />";
        $txt .= "frequenceJours = ".$even->recur_data."<br />";

        $date = $even->recur_enddate;
        $txt .= "frequenceDateFin = ";
        $txt .= $date->year."-".$date->month."-".$date->mday;
        $txt .= "<br />";

        $txt.="Alarme = ".$even->alarm."<br />";

        return $txt;

    }
}
```

Sauvegarde des événements

Enfin, pour sauvegarder l'événement, nous disposons de la fonction `mcal_store_event()`.

mcal_store_event()

Enregistre l'événement en cours. (À l'issue de cet appel, l'état de l'événement courant est indéterminé. Il convient donc de faire un appel à `mcal_event_init()`.)

Syntaxe	`int mcal_store_event(resource $idConnexion)`
`$idConnexion`	Identifiant de connexion tel que retourné par `mcal_open()`.
`retour`	Identifiant de l'événement ainsi ajouté, ou `FALSE` en cas d'échec.


REMARQUE

mcal_append_event()
Depuis la version 4.0, est apparue la fonction `mcal_append_event()`*, qui semble avoir le même comportement que* `mcal_store_event()` *(même si la documentation PHP indique que l'une sert à modifier un événement, tandis que l'autre sert à l'ajouter à l'agenda).*

Ceci nous permet d'enrichir notre objet *MCAL_Evenement* avec une méthode `sauve()`.

Listing 8.17 : mcal_evenement_inc.php (extrait)

```
<?php

// Extrait de MCAL_Evenement

class MCAL_Evenement {

// (...)

    // Sauve l'événement
    function sauve($mcal)
    {
        $this->prepare($mcal);
        $id=mcal_store_event($mcal);

        if ($id !== FALSE) {
            $this->id=$id;
        }
        return $id;
    }

}
```

Remarquez, qu'ici, nous en profitons pour stocker l'identifiant de l'événement dans l'objet *MCAL_Evenement*.

Maintenant, comme nous ne voulons pas avoir écrit cette classe pour rien, nous allons l'utiliser afin d'enregistrer un événement dans l'agenda. Mais, pour cela, un formulaire de saisie étant plus que pratique, nous allons ajouter à cette classe une méthode permettant d'afficher un formulaire adapté.

Listing 8.18 : mcal_evenement_inc.php (extrait)

```
<?php

// extrait de MCAL_Evenement

class MCAL_Evenement {

// (...)

    /**
```

```
 * Affiche un formulaire de saisie d'événement
 *
 * @param action string page à appeler lors de l'envoi des données
 */
function afficheFormulaire($action="mcal_evenement_sauve.php")
{
    echo "<form action=\"$action\" method=\"post\">\n";
    echo "<table border=\"0\" cellspacing=\"0\" cellpadding=\"2\">\n";
    echo "<tr class=\"champsdescription\"><td><b>Titre</b></td>";
    echo "<td><input type=\"texte\" name=\"titre\"/></td>";
    echo "<td><select name=\"visibilite\">";
    echo "<option value=\"0\">Privé</option>";
    echo "<option value=\"1\">Publique</option>";
    echo "</select></td></tr>\n";
    echo "<tr class=\"champsdescription\"><td><b>Catégorie</b></td>";
    echo "<td colspan=\"2\"><select name=\"categorie\">";
    echo "<option value=\"boulot\">Boulot</option>";
    echo "<option value=\"loisirs\">Loisirs</option>";
    echo "<option value=\"divers\">Divers</option>";
    echo "</select></td>";
    echo "</tr>\n";
    echo "<tr class=\"champsdescription\"><td><b>Description</b></td>";
    echo "<td colspan=\"2\"><textarea name=\"description\">";
    echo "</textarea></td>";
    echo "</tr>\n";
    echo "<tr class=\"champsdescription\"><td><b>Salle</b>(*)</td>";
    echo "<td colspan=\"2\"><input type=\"text\" name=\"salle\"></td>";
    echo "</tr>\n";
    echo "<tr class=\"champsdate\"><td><b>Date Début</b></td>";
    echo "<td><input type=\"text\" name=\"dateDebut\" /></td>";
    echo "<td>(AAAA-MM-JJ)</td>";
    echo "</tr>\n";
    echo "<tr class=\"champsdate\"><td><b>Heure Début</b></td>";
    echo "<td><input type=\"text\" name=\"heureDebut\" /></td>";
    echo "<td>(HH:MM:SS)</td>";
    echo "</tr>\n";
    echo "<tr class=\"champsdate\"><td><b>Date Fin</b></td>";
    echo "<td><input type=\"text\" name=\"dateFin\" /></td>";
    echo "<td>(AAAA-MM-JJ)</td>";
    echo "</tr>\n";
    echo "<tr class=\"champsdate\"><td><b>Heure Fin</b></td>";
    echo "<td><input type=\"text\" name=\"heureFin\" /></td>";
    echo "<td>(HH:MM:SS)</td>";
    echo "</tr>\n";
    echo "<tr class=\"champsfrequence\">";
    echo "<td><b>Fréquence</b></td>";
    echo "<td>1 <select name=\"frequenceType\">";
    echo "<option value=\"0\">(Evénement ponctuel)</option>";
    echo "<option value=\"1\">Jour</option>";
    echo "<option value=\"2\">Semaine</option>";
    echo "<option value=\"3\">Mois (même date)</option>";
    echo "<option value=\"4\">Mois (même jour,semaine)</option>";
```

```
            echo "<option value=\"5\">An</option>";
            echo "</select></td>\n";
            echo "<td>sur <input type=\"text\" name=\"frequence\"".
                 " size=\"3\"></td>";
            echo "</tr>\n";
            echo "<tr class=\"champsfrequence\">";
            echo "<td><b>Jours de la semaine</b><br />(si hebdomadaire)</td>";
            echo "<td colspan=\"2\"><table><tr>";
            echo "<td><input type=\"checkbox\" name=\"dimanche\" />Dimanche</td>";
            echo "<td><input type=\"checkbox\" name=\"lundi\" />Lundi</td>";
            echo "<td><input type=\"checkbox\" name=\"mardi\" />Mardi</td>";
            echo "<td><input type=\"checkbox\" name=\"mercredi\" />Mercredi</td>";
            echo "</tr><tr>";
            echo "<td><input type=\"checkbox\" name=\"jeudi\" />Jeudi</td>";
            echo "<td><input type=\"checkbox\" name=\"vendredi\" />Vendredi</td>";
            echo "<td><input type=\"checkbox\" name=\"samedi\" />Samedi</td>";
            echo "</tr></table></td></tr>\n";
            echo "<tr class=\"champsfrequence\">";
            echo "<td><b>Jusqu'au</b></td>";
            echo "<td><input type=\"text\" name=\"frequenceDateFin\" /></td>";
            echo "<td>(AAAA-MM-JJ)</td></tr>\n";
            echo "<tr class=\"champsalarme\"><td><b>Prevenez moi</b></td>";
            echo "<td colspan=\"2\">";
            echo "<select name=\"alarme\">";
            echo "<option value=\"0\">(pas)</option>";
            echo "<option value=\"5\" selected=\"selected\">5 min.</option>";
            echo "<option value=\"10\">10 min.</option>";
            echo "<option value=\"15\">1/4 h.</option>";
            echo "<option value=\"30\">1/2 h.</option>";
            echo "<option value=\"60\">1 h.</option>";
            echo "</select>";
            echo " avant</td></tr>";
            echo "<tr><td colspan=\"3\">";
            echo "(*) Exemple d'attribut personnalisé</td></tr>\n";
            echo "<tr><td colspan=\"3\" align=\"center\">";
            echo "<input type=\"submit\" value=\"Ajouter\" />";
            echo "</td></tr>";
            echo "</table>\n";
            echo "</form>";
        }

// (...)

}
```

Ce formulaire aura l'allure suivante (certes, il manque de fioritures, mais c'est efficace) (voir fig. 8.7) :

Lors de la validation du formulaire, la page *mcal_evenement_sauve.php* sera appelée. Celle-ci a donc pour mission de copier les paramètres issus de la méthode `POST` dans un objet *MCAL_Evenement* et d'appeler tout bonnement la méthode `sauve()` après avoir ouvert une connexion sur l'agenda.

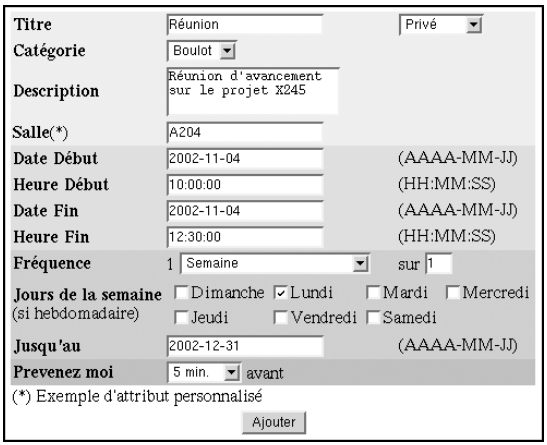


Figure 8.7 : *Formulaire de saisie d'événement*

Listing 8.19 : mcal_evenement_sauve.php

```
<?php
    include("mcal_03_inc.php");
    include("mcal_evenement_inc.php");

    // Connexion à l'agenda
    $mcal = MCAL_Agenda::connect();


    // Instanciation d'un objet MCAL_Evenement
    // (objet construit par nos soins)

    $evenement = new MCAL_Evenement();

    // Copie des données issues du formulaire
    // dans la structure de l'objet

    $evenement->visibilite    = $_POST["visibilite"];
    $evenement->titre         = stripslashes($_POST["titre"]);
    $evenement->categorie     = $_POST["categorie"];
    $evenement->description   = stripslashes($_POST["description"]);

    $evenement->monAttribut1  = $_POST["salle"];

    $evenement->dateDebut     = $_POST["dateDebut"];
    $evenement->heureDebut    = $_POST["heureDebut"];

    $evenement->dateFin       = $_POST["dateFin"];
    $evenement->heureFin      = $_POST["heureFin"];

    $evenement->frequenceType = $_POST["frequenceType"];
    $evenement->frequence     = $_POST["frequence"];

    if ($evenement->frequenceType == 2) {
```

```
            $evenement->frequenceJours = 0;
            if ($_POST["dimanche"] == "on")
                                $evenement->frequenceJours += 1;
            if ($_POST["lundi"] == "on")
                                $evenement->frequenceJours += 2;
            if ($_POST["mardi"] == "on")
                                $evenement->frequenceJours += 4;
            if ($_POST["mercredi"] == "on")
                                $evenement->frequenceJours += 8;
            if ($_POST["jeudi"] == "on")
                                $evenement->frequenceJours += 16;
            if ($_POST["vendredi"] == "on")
                                $evenement->frequenceJours += 32;
            if ($_POST["samedi"] == "on")
                                $evenement->frequenceJours += 64;
        }
        $evenement->frequenceDateFin = $_POST["frequenceDateFin"];

        $evenement->alarme        = $_POST["alarme"];

        $id = $evenement->sauve($mcal);

    ?>
    <html>
    <body>
    <h1>Enregistrement d'un événement</h1>
    <?php
        if ($d !== FALSE) {
            echo "Votre nouvel événement a été enregistré (avec l'identifiant".
                 " $id)";
        } else {
            echo "La tentative d'enregistrement de l'événement à échoué";
        }
        echo $evenement->toHTMLString();
    ?>
    </body>
    </html>
```


REMARQUE

Utilisation de stripSlashes()

Comme les paramètres `titre` *et* `description` *sont des chaînes de caractères librement saisies par l'utilisateur, ils peuvent contenir (entre autres) des apostrophes. Or, par défaut, PHP est configuré avec l'option* `magic_quotes_gpc = on`*, ce qui fait que les valeurs passées par les méthodes* `POST`*, et* `GET` *ou par cookies voient leurs apostrophes précédées d'un anti-slash (comme le fait la fonction* `addSlashes()`*). Il faut les supprimer, car, bien que fort utiles dans le cas d'un ajout dans une base de données, ils sont gênants lorsque les valeurs sont simplement passées en paramètre d'une fonction (ici les fonctions* `mcal`*).*

Opérations sur les événements enregistrés

Une fois l'événement enregistré, il est tout de même possible de supprimer facilement l'alarme qui lui est associée par un appel à la fonction `mcal_snooze()`.

mcal_snooze()

Supprime l'alarme associée à un événement.

Syntaxe	`boolean mcal_snooze(resource $idConnexion, int $idEvenement)`
`$idConnexion`	Identifiant de connexion tel que retourné par `mcal_open()`.
`$idEvenement`	Identifiant de l'événement.
`retour`	`TRUE`.

La bibliothèque propose une fonction qui est censée retourner la liste des événements ayant une alarme programmée à la date donnée. Mais il est difficile d'en tirer quelque chose. En voici tout de même la syntaxe (corrigée par rapport à celle proposée par la documentation PHP) :

```
array mcal_list_alarms(resource $idConnexion[, int $annee [, int $mois [,
int $jour [, int $heure [, int $minutes [, int $secondes]]]]]])
```

Et, bien évidemment, il est possible de supprimer un événement de l'agenda par un appel à `mcal_delete_event()`.

mcal_delete_event()

Supprime un événement.

Syntaxe	`boolean mcal_delete_event(resource $idConnexion, resource $idEvenement)`
`$idConnexion`	Identifiant de connexion tel que retourné par `mcal_open()`.
`$idEvenement`	Identifiant de l'événement à supprimer.
`retour`	`TRUE` (même si l'événement n'existe pas).

Divers

La bibliothèque `MCAL` présente des fonctions destinées à la manipulation des agendas (dans leur ensemble). Il s'agit des fonctions `mcal_create_calendar()`, `mcal_rename_calendar()`, `mcal_delete_calendar()`, mais celles-ci ne semblent pas avoir été véritablement implémentées et se contentent de retourner `TRUE`.

Chapitre 9

La gestion des fichiers et des répertoires

La manipulation et le traitement des données nécessitent, voire imposent, de pouvoir stocker celles-ci une fois le travail et les différentes opérations effectués. Même si, souvent, l'utilisation d'une base de données est plus efficace, le stockage sur le disque reste une solution simple à mettre en œuvre, et parfois même la seule solution pertinente. Dans ce chapitre, nous allons voir comment le langage PHP peut permettre de lister, lire et écrire des fichiers.

Tout au long de ce chapitre, nous allons mettre en place différentes fonctions qui nous permettront, au final, de développer un explorateur de fichiers simple et sympa (attention toutefois : il n'est pas à laisser entre toutes les mains).

Dans une deuxième partie, nous allons voir comment il est possible d'accéder au système de fichier, réseau, socket à l'aide d'une méthode introduite avec PHP 4.3 et disponible également sur PHP 5 : Les streams ou flux.

9.1. Le système de fichiers POSIX

Même si vous êtes un adepte des produits Microsoft, il est probable que votre application sera hébergée sur une plateforme de type UNIX/Linux. C'est en effet le système d'exploitation adopté par la majorité des hébergeurs – qu'ils soient ou non gratuits. Nous vous offrons donc, ici, un petit cours de rattrapage en faisant un point sur les spécificités du système de fichiers de ces systèmes d'exploitation (ceci est valable pour tous les systèmes de fichiers à la norme POSIX que sont Linux et tous les UNIX modernes).

Même si, petit à petit, Windows tente (péniblement) de rattraper son retard en termes de sécurité, avec UNIX/Linux, les droits des utilisateurs ne sont déjà pas un vain mot. Ainsi, l'utilisateur devra nécessairement s'identifier pour se connecter au système, ce qui constitue la base de la sécurité de la machine.

UNIX étant, dès son origine, un système multi-utilisateur, il a été conçu de sorte que les fichiers ne soient accessibles qu'à certains utilisateurs et certains groupes d'utilisateurs. Seul l'administrateur, appelé le plus souvent "`root`", possède tous les droits sur tous les fichiers du système (s'il existait des virus dans un environnement UNIX/Linux, il faudrait encore qu'ils accèdent au compte "`root`" pour causer des dommages irréparables ; de même, un utilisateur protégeant ses fichiers vis-à-vis des autres utilisateurs ne pourrait être contaminé).

Chaque utilisateur peut appartenir à un ou plusieurs groupes que l'on définira librement (ex. : groupe des chercheurs, des invités, des utilisateurs du service web, etc). Chaque fichier est la propriété d'un utilisateur et d'un groupe d'utilisateurs.

Ainsi, si vous listez le contenu d'un répertoire d'un système UNIX/Linux (si vous faites appel aux services d'un hébergeur, vous pouvez, par exemple, vous connecter sous votre compte FTP, et taper la commande `ls`), vous observerez une série de lignes ayant l'allure suivante :

```
-rwxr-wr-- 1 laurent  user 985540 Mai 4 09:16 Monfichier
```

Chaque ligne est associée à un fichier ou un répertoire dont le nom est précisé dans la dernière colonne (ici : *Monfichier*).

On ne rigole pas avec la casse...
UNIX respecte la casse. Faites donc bien attention au respect des majuscules et minuscules des noms de fichiers et répertoires. Vous pouvez très bien avoir un fichier nommé toto.txt qui coexiste, dans le même répertoire, avec un fichier toto.TXT.

Les troisième et quatrième colonnes nous indiquent que le propriétaire du fichier est "`laurent`" et que le groupe auquel appartient le fichier est "`user`".

Mais attention, ce n'est pas parce que le fichier appartient à "`laurent`" et au groupe "`user`" que Laurent et les membres du groupe "`user`" ont tous les droits, ni non plus que tous les autres utilisateurs n'ont aucun droit sur ce fichier.

Les permissions sont définies ici par les propriétés *-rwxrw-r--* qu'il faut décomposer en un caractère suivi de trois blocs de trois caractères. Nous reviendrons plus tard sur ce premier caractère. Le premier bloc de trois caractères précise les droits de l'utilisateur, le bloc suivant ceux des membres du groupe et enfin le dernier bloc de trois caractères indique les droits des autres utilisateurs.

Chaque bloc se décompose ainsi :

1. Un premier caractère pouvant prendre la valeur "r" pour indiquer que le fichier est accessible en lecture (pour l'utilisateur, les membres du groupe ou les autres, selon les cas) ou "-" sinon. Dans le cas d'un répertoire, ceci indique que l'on peut lister le contenu du répertoire.
2. Un second caractère pouvant prendre la valeur "w" pour indiquer que le fichier est accessible en écriture (pour l'utilisateur, les membres du groupe ou les autres, selon les cas), ce qui inclut la possibilité d'effacer le fichier ou "-" sinon. Dans le cas d'un répertoire, ceci indique que l'on peut modifier les fichiers contenus dans le répertoire (à condition que les droits individuels des fichiers le permettent).
3. Un troisième caractère pouvant prendre la valeur "x" pour indiquer que le fichier peut être exécuté (par l'utilisateur, les membres du groupe ou les autres, selon les cas) à condition, toutefois, que ce soit un "exécutable" ou "-" sinon. Dans le cas d'un répertoire, ceci indique que l'on peut se déplacer dans le répertoire.

Dans notre exemple, nous pouvons donc dire que le fichier est accessible en lecture, écriture et exécution pour l'utilisateur. Le groupe possède les permissions en lecture et écriture, mais ne peut pas exécuter le fichier. Enfin, les autres utilisateurs ne sont autorisés qu'à la lecture.

Les droits ne sont généralement pas spécifiés sous forme alphabétique, mais sous la forme d'un nombre exprimé en octal (base 8) et composé de trois chiffres (précédés d'un 0 pour indiquer à PHP qu'il s'agit d'un nombre en octal et non en décimal). Chaque chiffre représente les droits d'une catégorie (propriétaire, groupe et autres). Pour chaque catégorie, il faut faire la somme des valeurs de chaque droit en considérant que r=4, w=2 et x=1.

Imaginons, pour prendre un exemple, un fichier possédant les droits `rwx` pour l'utilisateur, `r-x` pour le groupe et `r--` pour toutes les autres personnes.

Tableau 9.1 : Les droits en octal donnent pour le cas présent : 0754

	Propriétaire	Groupe	Autres
`Read(r)` = 4	Oui	Oui	Oui
`Write(w)` = 2	Oui	Non	Non
`Execute(x)` = 1	Oui	Oui	Non
Total	4 + 2 + 1 = 7	4 + 0 + 1 = 5	4 + 0 + 0 = 4

Retenez bien cette leçon, nous nous en resservirons un peu plus tard.

Revenons maintenant au premier caractère de la chaîne indiquant les droits du fichier. Il précise le type du "fichier". Ici, "`-`" indique que c'est un fichier classique, mais cela aurait pu être un répertoire représenté par la lettre "`d`", un périphérique défini par "`b`" ou "`c`", ou un encore un lien symbolique associé à la lettre "`l`".

Vous savez tous ce qu'est un fichier ou un répertoire (même si certains ont réussi à imposer leur terminologie et parlent de dossier). Afin de ne pas nous étendre davantage sur les spécificités Linux/UNIX, nous ne parlerons pas des périphériques. Mais, afin de mieux comprendre certaines fonctions proposées par PHP, il est bon de vous présenter ce qu'est un lien.

Sous Linux/UNIX, il est possible de créer un "fichier" (une entrée dans un répertoire) qui sera en fait un lien, dit lien symbolique, vers un "vrai" fichier. Sous Windows, vous connaissez un ersatz du lien symbolique appelé "raccourci". Double-cliquer sur un raccourci Windows revient à double-cliquer sur le fichier pointé. En revanche, si vous souhaitez accéder au contenu du raccourci, vous n'accéderez qu'à un fichier descripteur et non au contenu du fichier pointé. Sous Linux/UNIX, que vous accédiez au lien symbolique ou directement au fichier, le résultat est le même.

Il existe une variante du lien symbolique appelé lien physique, mais la nuance est subtile. Alors que le lien symbolique est un renvoi vers le fichier, le lien physique est une copie du descripteur de fichier. De ce fait, contrairement au lien physique, si vous supprimez le fichier "original", le lien symbolique n'est plus valide.

Dans les lignes retournées par la commande `ls` (via FTP ou `ls -l` en accès direct sur le système), le chiffre suivant correspond au nombre de liens physiques vers le fichier. Dans notre exemple, 1 signifie que le fichier ne possède aucun lien mis à part lui-même.

Le nombre qui suit le groupe d'appartenance du fichier est le poids (la taille) en octets. Ensuite, on retrouve la date de la dernière modification du fichier.


REMARQUE

Point et point-point sont dans un répertoire...
Pour chaque répertoire vous noterez deux fichiers portant des noms particuliers : il s'agit de "`.`" et "`..`". Point est, en fait, une représentation du répertoire courant, et point-point une représentation du répertoire supérieur. C'est un point ;-) à connaître lorsque vous souhaitez en lister le contenu.

Les fichiers textes, les fichiers binaires

Les fichiers peuvent être rangés en deux catégories : les fichiers textes et les fichiers binaires. Dans le premier cas, ce sont des fichiers qui ne contiennent que des lignes de texte ; ce peut être un document quelconque, un listing de programme ou des données textes. Le fichier binaire est destiné à contenir des données brutes qui ne peuvent pas être sectionnées en lignes. Ce sont, en règle générale, des fichiers de type image ou des données complexes comme des fichiers de traitement de texte ou de tableur.

Sous un système de type UNIX, la différence entre les fichiers textes et binaires concerne les traitements de fin de fichier et de fin de lignes. Alors qu'un fichier texte est séparé en lignes terminées par un "`\n`" et que la fin du fichier est identifiée par un "`\0`", la fin d'un fichier de type binaire est donnée par le système d'exploitation lui-même. C'est sa taille en octets qui indique que le fichier a été entièrement lu et, donc, que le pointeur de lecture est arrivé à la fin du fichier. Il est, de ce fait, inutile de spécifier le mode binaire dans vos fonctions d'ouverture de fichier.

UNIX, conforme à la norme POSIX, ne fait aucune différence entre un fichier binaire et un fichier texte.

En revanche, lorsque vous manipulez un fichier avec Windows, il est nécessaire de préciser si le fichier à ouvrir est de type texte ou binaire. Par défaut, il sera considéré comme étant de type texte.

Fins de lignes

Attention également aux fins de lignes sous Windows. Toutes les lignes sont séparées par le couple de caractères '`\r\n`', qui indique de passer à la ligne suivante. Lorsque vous passez un fichier d'un système à un autre, n'hésitez pas à effectuer une conversion du fichier texte à l'aide de la commande UNIX `dos2unix` ou depuis votre éditeur s'il en possède l'option (c'est le cas d'`UltraEdit` par exemple). Ce détail, qui peut paraître insignifiant, peut se révéler source de problèmes dans bien des cas.

9.2. Accéder au système de fichiers du serveur

Lire et écrire le contenu d'un fichier

La méthodologie pour traiter un fichier est la suivante :

- Ouverture du fichier ;
- Opération sur le fichier (lecture, écriture, lecture et écriture) ;
- Fermeture du fichier.

Ouvrir et fermer un fichier

L'ouverture d'un fichier se fait à l'aide de l'instruction `fopen()`. L'instruction retourne un descripteur de fichier qui servira par la suite lors des opérations d'entrée et de sortie. Il est nécessaire de donner à la fonction le mode d'ouverture (lecture, écriture, lecture et écriture, etc.) du fichier. Notez que cette fonction permet aussi d'ouvrir des "fichiers" distants en donnant une URL à la place du chemin du fichier. Notez que vous ne pouvez pas utiliser les ouvertures de fichiers à distance si vous avez compilé PHP avec l'option `--disable-url-fopen-wrapper` pour PHP 4.0.3, ou initialisé le paramètre `allow_url_fopen` à `off` dans le fichier *php.ini* pour les versions supérieures de PHP.

fopen()

Ouverture d'un fichier ou d'une URL.

Syntaxe

```
resource fopen(string $nomFichier, string $mode [, int
$cheminInclude [, resource $contexte]])
```

`$nomFichier`	Chemin du fichier, URL (HTTP ou FTP) du fichier ou encore entrée/sortie standard.
`$mode`	Définit le mode d'ouverture du fichier : "`r`" pour la lecture seule (le fichier doit exister). "`r+`" pour une lecture et une écriture (le fichier doit exister). "`w`" pour une écriture avec écrasement des données. "`w+`" pour un mode en lecture et écriture avec écrasement des données (si le fichier n'existe pas, il sera créé). "`a`" pour l'ouverture en ajout de données. "`a+`" pour l'ouverture du fichier en mode lecture et écriture par ajout (si le fichier n'existe pas, il sera créé). Sous Windows, vous pouvez accoler l'un des arguments suivants : "`t`" (valeur par défaut) signalant que le fichier est un fichier texte. "`b`" qui désigne un fichier binaire.
`$cheminInclude`	`TRUE` si le fichier doit être recherché dans les répertoires précisés dans `include_path`., `FALSE` sinon. La valeur par défaut est `FALSE`.
`$contexte`	Paramètre optionnel, utilisable depuis les versions 4.3 de PHP, spécifiant la ressource de contexte créée par `stream_context_create()` à utiliser. Voir le sous-chapitre sur les flux pour plus de renseignements sur les contextes de flux.
`retour`	Retourne un pointeur sur le fichier, ou `FALSE` en cas d'erreur.

include_path
La variable `include_path` est définie dans le php.ini et spécifie à PHP une liste de dossiers où les fonctions comme `include()` ou `require()` vont chercher les fichiers.

Vous pouvez vous reporter au chapitre "Prise en main" pour plus de détails sur le fichier php.ini.

Le nom du fichier peut être indiqué sous différentes formes :

- Un chemin de fichier ; `fopen()` tentera alors d'accéder à ce fichier dans le mode demandé.
- Une URL du style **http://www.toto.com** et, dans ce cas, une connexion HTTP est ouverte sur le serveur. L'identifiant retourné pointe sur le document. Notez que vous ne pouvez ouvrir ces fichiers qu'en lecture seule. Vous pouvez également accéder à des fichiers sur un serveur demandant une autorisation d'accès. Pour cela, vous devez passer, dans l'URL, l'identifiant de l'utilisateur ainsi que son mot de passe de la façon suivante : **http://utilisateur:motdepasse@domaine.com**.
- Une adresse FTP. Une connexion FTP est créée avec le serveur. Vous pouvez ouvrir les fichiers en lecture ou en écriture. Les doubles modes lecture et écriture (mode full duplex) ne sont pas supportés. Pour l'authentification, utilisez la syntaxe identique à une

authentification HTTP : **ftp://utilisateur:motdepasse@ftp.domaine.com**. Notez que le serveur distant doit supporter le mode passif ; dans le cas contraire la connexion ne pourra s'effectuer.

- Une entrée/sortie standard du style `php://`. Ce peut-être `php://stdin` pour l'entrée standard, `php://stdout` pour la sortie standard ou `php://stderr` pour la sortie d'erreur.

Les actions à effectuer sur le fichier sont à définir dès son ouverture. C'est le mode du fichier qui indique à la fonction `fopen()` pour quelle opération le fichier est ouvert.

```
<?php
$fp = fopen("/home/laurent/data.txt", "a");
// Ouverture du fichier data.txt en ajout de données
$fp = fopen("http://www.linux.org/", "r");
// Ouverture de l'URL distante http://www.linux.org
$fp = fopen("http://utilisateur:motdepasse@localhost/", "r");
// Ouverture d'une URL demandant une authentification
$fp = fopen("ftp://utilisateur:motdepasse@ftp.tuxfamily.org/", "w");
// Ouverture d'une connexion FTP avec l'utilisateur et le mot de passe définie
?>
```

La partie du nom de fichier du genre `http://`, `ftp://`, `php://` fait référence à un gestionnaire de flux (les streams). Il en existe d'autres. Nous verrons en particulier celui chargé de la gestion des fichiers compressés et nous verrons également qu'il est possible de définir ses propres gestionnaires.


RENVOI

Pour plus d'informations, reportez-vous plus loin au chapitre « Les steams ou les flux »


REMARQUE

Droits de propriété...
N'oubliez pas qu'UNIX gère les droits de lecture et d'écriture. Si vous utilisez FTP pour uploader un fichier sous votre compte utilisateur (avec les droits qui lui sont propres), le serveur HTTP n'aura pas forcément les permissions pour y accéder (en lecture ou en écriture). Vous devrez alors modifier les droits via votre accès FTP et la commande `chmod`. Par exemple, pour donner tous les droits :

```
chmod 777 monfichier
```


ATTENTION

Le chemin des fichiers sous Windows
Sous Windows, le caractère habituellement utilisé pour séparer les noms des répertoires est l'anti-slash ; assurez-vous bien de doubler ces anti-slashes.

```
<?php
$fp = fopen("c:\\Rep1\\Rep2\\data.txt", "a");
?>
```

Fichiers HTTP

Pour les fichiers accédés via HTTP, l'en-tête retourné par le serveur ne fait pas partie des données accessibles à l'aide de l'identifiant de fichier. Il est toutefois possible d'y accéder via la variable globale `$http_response_header` (attention, cette variable est bien en minuscules et non en majuscules). Celle-ci contient alors un tableau indexé où chaque entrée est une ligne de l'en-tête.

À partir de PHP 4.3, vous devez utiliser la fonction `file_get_wrapper_data()()` pour récupérer cette information.

file_get_wrapper_data()

Retourne l'en-tête HTTP d'un fichier ouvert (à partir de PHP 4.3).

Syntaxe	`array file_get_wrapper_data(resource $fp)`
`$fp`	Pointeur du fichier tel que retourné par `fopen()`.
`retour`	Tableau indexé où chaque entrée est une ligne de l'en-tête.

Fichiers temporaires

En supposant que vous ayez plusieurs fichiers à manipuler à la fois, il est probable que vous soyez amené à utiliser des fichiers temporaires. Vous pouvez bien entendu créer ces fichiers avec la fonction `fopen()` et les supprimer une fois que la manipulation a été effectuée avec l'instruction `unlink()`, mais le langage PHP possède une fonction adéquate pour gérer ces fichiers. La commande `tmpfile()` ouvre un fichier sous un nom unique et retourne un pointeur. Le fichier est automatiquement détruit lorsque la ressource est libérée avec l'instruction `fclose()`.

tmpfile()

Créer un fichier temporaire en mode écriture seule et retourne un identifiant unique permettant d'écrire dans ce fichier.

Syntaxe	`resource tmpfile(void)`
`retour`	Retourne un pointeur de fichier.

```
<?php
$fpT = tmpfile();
//  Traitements divers
fclose($fpT);
?>
```

Fermeture des fichiers

La fermeture des fichiers ainsi ouverts se fait à l'aide de la fonction `fclose()`. La fonction a pour objectif de libérer les ressources utilisées. Attention, si vous oubliez de fermer vos fichiers, vous risquez de perdre des données.

fclose()

Ferme un fichier préalablement ouvert.

Syntaxe	`boolean fclose(resource $fp)`
`$fp`	Pointeur du fichier tel que retourné par `fopen()`, `tmpfile()` ou `fsockopen()`.
`retour`	`TRUE` en cas de réussite, `FALSE` sinon.

Si le pointeur de fichier n'est pas valide, la fonction renvoie une erreur.

```
<?php
$fp = fopen("monfichier.txt", "a");
//  Traitements divers
if (fclose($fp)) {
    echo "Le fichier monfichier.txt a été correctement fermé !";
} else {
    echo "Le fichier monfichier.txt n'a pas été fermé !";
}
?>
```

`fsockopen()` est une fonction d'ouverture d'une socket de connexion. Rendez-vous dans le chapitre "La gestion des protocoles" pour plus d'informations sur le sujet.

Écrire dans un fichier

Pour écrire dans un fichier, vous disposez de la fonction `fwrite()`.

fwrite()

Écriture d'une chaîne de caractères dans un fichier.

Syntaxe	`int fwrite(resource $fp, string $chaine [, int $longueur])`
`$fp`	Identifiant de fichier tel que retourné par `fopen()`, `popen()`, `tmpfile()` ou `fsockopen()`.
`$chaine`	Chaîne de caractères à écrire dans le fichier.

`$longueur`	Paramètre optionnel indiquant combien de caractères doivent être écrits. S'il est omis, la chaîne complète est écrite dans le fichier.
`retour`	Nombre de caractères qui ont été écrits dans le fichier. En cas d'erreur, l'instruction retourne `FALSE`.

La fonction `fwrite()` dispose d'un alias appelé `fputs()`. Les deux fonctions sont donc utilisées de la même façon.

Écriture binaire

Notez bien que la fonction `fwrite()` n'excepte qu'une chaîne de caractères comme paramètre. Si vous souhaitez écrire des données binaires, vous devrez convertir au préalable la valeur de chaque octet en son équivalent ASCII.

Ainsi, `fwrite($fp, 255);` stockera la chaîne de caractères "255" et non un octet ayant la valeur 255 (comme on pourrait s'y attendre).

Pour ajouter au fichier l'octet ayant la valeur 255, il faut utiliser l'appel suivant : `fwrite($fp, chr(255));`.

Les quelques fonctions vues jusque-là nous permettent d'écrire un petit programme qui pourrait constituer la base d'un système de discussion.

En effet, ce script écrit, dans un fichier texte, des messages envoyés à l'aide d'un formulaire HTML. Le fichier résultant possède une ligne par message et contient la date du message, le pseudo de l'utilisateur, et le message lui-même, les éléments étant séparés par une tabulation. Nous pourrons donc, par la suite, afficher la liste des derniers messages qui ont été écrits dans le fichier texte.

Listing 9.1 : fwrite.php

```
<?php
//  Vérifie que les données sont bien postées
if ($_POST["message"] && $_POST["pseudo"])
{
    //  Ouverture du fichier en écriture (mode ajout)
    $fp = fopen("message.txt", "a+");
    //  Récupération de l'heure
    $heure = date('H:m:s');
    //  Ecriture de l'heure et d'une tabulation
    fwrite($fp, $heure."\t");
    //  Ecriture du pseudo et d'une tabulation
    /*
        On utilise l'instruction htmlEntities afin de
        convertir les caractères spéciaux dans leur version
        HTML
    */
    fwrite($fp, htmlEntities($_POST["pseudo"])."\t");
    //  Ecriture du message
    fwrite($fp, htmlEntities($_POST["message"])."\n");
    //  On ferme le fichier.
    fclose($fp);
```

```
}
?>

<html>
<title>Ecrire des messages dans un fichier</title>
</head>
<body>
<table border="0" width="100%">
    <tr>
    <form method="post">
    <td width="600">
        pseudo:
        <input type="text" name="pseudo"
               value="<?php htmlEntities($_GET["pseudo"]); ?>"
               size="20" maxlength="20">
        message:
        <input type="text" name="message" size="40" maxlength="255">
        <input type="submit" name="envoyer" VALUE="&gt;&gt;">
    </td>
    </form>
    <td> </td>
    </tr>
</body>
</html>
```

Le fichier résultant présente alors cette forme :

```
15:07:48<tab>Laurent<tab>C'est bon, je suis enfin en vacances !
15:07:50<tab>Damien<tab>Non mais tu r&ecirc;ves... tu as un chapitre &agrave;
✂ rendre !
15:07:49<tab>Pem<tab>Hi hi hi... :)
15:07:48<tab>Laurent<tab>Groupf... Je vais devoir y passer la nuit l&agrave; !
15:07:49<tab>Thomas<tab>Tu devrais faire comme moi.
15:07:00<tab>Thomas<tab>Des jours que je ne dors plus !
15:07:10<tab>Thomas<tab>Mon secret ? La caf&eacute;ine !
```

Il est temps de passer à la lecture du contenu d'un fichier.

Lire des informations dans un fichier

La lecture des données contenues dans un fichier peut se faire octet par octet, bloc de N octets par bloc de N octets, ligne par ligne ou encore d'un bloc.

Ceci est assuré par les différentes fonctions que sont `fread()`, `fgets()`, `fgetss()`, `fgetc()` et deux instructions particulières que sont `file()` et `readfile()`.

Lecture octet par octet

Tout comme le langage C, PHP possède la fonction `fgetc()` qui retourne le caractère se trouvant à la position courante du pointeur de fichier. Notez qu'en langage C ce n'est pas le caractère qui est retourné, mais son code ASCII.

fgetc()

Retourne le caractère se trouvant à la position courante du pointeur de lecture.

Syntaxe	`string fgetc(resource $fp)`
`$fp`	Identifiant de fichier tel que retourné par `fopen()`, `popen()` ou `fsockopen()`.
`retour`	Le caractère se trouvant à la position courante, ou `FALSE` si c'est la fin du fichier.

Lecture N octets par N octets

`fread()` a pour objectif de lire des données dans un fichier, la lecture se faisant paquet d'octets par paquet d'octets (ce qui est habituellement utilisé pour les fichiers binaires ou pour ceux formatés sur la base de chaînes de caractères de longueur constante).

fread()

Lecture des données contenues dans un fichier.

Syntaxe	`string fread(resource $fp, int $nbOctet)`
`$fp`	Identifiant de fichier tel que retourné par `fopen()`, `popen()` ou `fsockopen()`.
`$nbOctet`	Nombre d'octets qui doivent être lus.
`retour`	Chaîne contenant les données lues, ou `FALSE` si c'est la fin du fichier.

Pour lire le contenu du fichier précédemment généré, nous pourrions donc envisager d'écrire le script suivant :

Listing 9.2 : fread.php

```
<?php
//  Reprenons notre fichier message.txt.
$fp = fopen("message.txt","r");
//  Lecture du fichier jusqu'à la fin de celui-ci.
while ($lecture = fread($fp, 70))
{
    //  Affichage des 70 caractères lus puis
    //  retour à la ligne.
    echo $lecture."<br />";
}
fclose($fp);
?>
```

Ce qui donnerait :

```
22:06:38 Laurent C'est bon, je suis enfin en vacances ! 22:07:10 Damie
n Non mais tu rêves... tu as un chapitre à rendre ! 22:07
:26 Pem Hi hi hi... :) 22:08:01 Laurent Groupf... Je vais devoir y pas
ser la nuit là ! 22:08:30 Thomas Tu devrais faire comme moi. 22
:08:30 Thomas Des jours que je ne dors plus ! 22:08:39 Thomas Mon secr
et ? La caféine !
```

Certes, tout le contenu y est (en HTML, les tabulations apparaissent comme une simple espace). Mais pour la mise en forme... ce n'est pas ça ! En fait, plutôt que de lire le fichier 70 caractères par 70 caractères, il aurait été préférable de le lire ligne par ligne. Mais `fread()` ne tient pas compte des retours à la ligne. Nous vous rassurons tout de suite, nous allons trouver une solution !


REMARQUE

Lire en une fois

Il est possible de lire un fichier texte en une seule fois simplement en passant en argument de taille la taille du fichier texte. Vous devrez alors utiliser l'instruction `fileSize()` qui nous retourne la taille du fichier en octets (cette fonction est décrite plus loin dans ce chapitre).

```
<?php
//  Ouverture du fichier message.txt.
$fp = fopen("message.txt","r");
//  Récupération de la taille du fichier en octets.
$taille = fileSize("message.txt");
//  Lecture de la totalité du fichier.
$lecture = fread($fp, $taille);
//  Affichage du fichier.
echo $lecture;
fclose($fp);
?>
```

Lecture ligne par ligne

Comme vous l'attendiez, vous pouvez également lire le fichier ligne par ligne. La fonction `fgets()` permet de retourner la ligne courante (où se trouve le pointeur). La lecture de la ligne se termine lorsque le caractère retour chariot '`\n`' est trouvé, lorsque le nombre de caractères maximum est lu, ou lorsque la fin du fichier a été détectée.

fgets()

Lecture de la ligne courante du fichier.

Syntaxe `string fgets(resource $fp, int $nbOctet)`

`$fp`	Identifiant de fichier tel que retourné par `fopen()`, `popen()`ou `fsockopen()`.
`$nbOctet`	Nombre d'octets qui doivent être lus.
`retour`	Chaîne contenant les données lues de la ligne courante, ou `FALSE` si le pointeur est en fin de fichier.

La lecture du fichier peut alors se faire comme suit :

Listing 9.3 : fgets.php

```
<?php
//  Ouverture du fichier
$fp = fopen("message.txt","r");
//  Lecture de chaques ligne
//  jusqu'à la fin du fichier
while ($lecture = fgets($fp,255))
{
    //  Affichage de la ligne
    echo $lecture;
    echo "<br />";
}
//  Fermeture du fichier
fclose($fp);
?>
```

Dans ce cas, le résultat obtenu est bien celui escompté.

Lire un fichier ligne par ligne permet éventuellement de manipuler ces lignes avant de les afficher.

Des fichiers formatés

Un fichier est, le plus souvent, censé contenir une série de données selon un format particulier choisi par le développeur. Cela peut être une série de valeurs séparées par des tabulations comme pour notre exemple du système de discussion.

Si le format du fichier a été bien pensé, alors son analyse, et donc la récupération des données qu'il contient, se trouve grandement simplifiée par l'utilisation de l'instruction `fscanf()`. Cette fonction lit les caractères en provenance d'un fichier que vous aurez préalablement ouvert, et les traitent en fonction d'un schéma que vous aurez spécifié.

fscanf()

Récupère les données d'une ligne d'un fichier en fonction d'un format spécifié.

Syntaxe `mixed fscanf(resource $fp, string $format [, string &$variable1 [, ...]])`

`$fp`	Identifiant sur un fichier ouvert à l'aide de l'instruction `fopen()`.
`$format`	Format de la chaîne de caractères et des données à récupérer.
`$variable1, ...`	Variables dans lesquelles les valeurs seront copiées.
`retour`	Selon les cas : Le nombre de données qui ont été lues si des variables ont été passées en paramètre. Un tableau indexé contenant les différentes valeurs si les paramètres ont été omis. Lorsque le pointeur de lecture arrive à la fin du fichier, la fonction retourne `FALSE`.

RENVOI

Le format de la chaîne est identique au format qu'utilise la fonction `sscanf()`. Rendez-vous au chapitre "La manipulation des chaînes de caractères" pour plus de détails sur les formats.

Ainsi, il est aisé de récupérer nos données depuis le fichier *messages.txt* et d'afficher les différents messages, ainsi que la date et le pseudo de l'utilisateur ayant posté ce message. Ensuite, il ne reste plus qu'à afficher cela dans une page HTML.

Listing 9.4 : fscanf.php

```
<?php
//  fichier "fwrite.php"
//  Ecrire des messages dans un fichier texte

//  Vérifie que les données sont bien postées
if($_POST["message"] && $_POST["pseudo"])
{
    //  Ouverture du fichier en écriture
    $fp = fopen("message.txt", "a+");
    //  Récupération de l'heure
    $heure = date('H:m:s');
    //  Ecriture de l'heure
    fwrite($fp, $heure."\t");
    //  Ecriture du pseudo
    /*
        On utilise l'instruction htmlEntities afin de
        convertir les caractères spéciaux dans leur entité
        HTML
    */
    fwrite($fp, htmlEntities($_POST["pseudo"])."\t");
    //  Ecriture du message
    fwrite($fp, htmlEntities($_POST["message"])."\n");
    //  On ferme le fichier.
    fclose($fp);
}
?>
```

```
<html>
<title>Ecrire des messages dans un fichier</title>
</head>
<body>
<table border="0" width="100%">
<?php
//  Affiche les messages
$fp = fopen("message.txt","r");
while($donnee = fscanf($fp, "%s\t".
                            "%s\t".
                            "%[^\t]\n",
                            $heure, $pseudo, $message))
{
    echo "<tr>";
    echo "  <td><font color='#0000ff'>";
    echo $heure;
    echo "  </font></td>";
    echo "  <td><b>";
    echo $pseudo;
    echo "  </b></td>";
    echo "  <td>";
    echo $message;
    echo "  </td>";
    echo "</tr>";
}
fclose($fp);
?>
</table>
<hr />
<table border="0" width="100%">
    <tr>
    <form method="post">
    <td width="600">
        pseudo:
        <input type="text" name="pseudo"
               value="<?php htmlEntities($_POST["pseudo"]);?>"
               size="20" maxlength="20">
        message:
        <input type="text" name="message" size="40" maxlength="255">
        <input type="submit" name="envoyer" VALUE="&gt;&gt;">
    </td>
    </form>
    <td> </td>
    </tr>
</body>
</html>
```


REMARQUE

Les espaces qui ne passent pas...

Dans le cas où vous devez récupérer une donnée comportant une espace, il est indispensable de ne pas utiliser le formatage des données avec "%s". Vous devez en


REMARQUE

revanche utiliser les expressions régulières afin de préciser quels caractères peut contenir la chaîne ou, plus simplement, préciser quels caractères ne peut contenir la chaîne (ici, la tabulation qui a été choisie comme délimiteur).


RENVOI

Pour plus d'informations sur les expressions régulières, reportez-vous au chapitre "La manipulation des chaînes de caractères"

Lecture d'un bloc

Le langage PHP possède une fonction permettant d'effectuer cette opération sans avoir à gérer l'ouverture et la fermeture des fichiers. L'instruction `file()` retourne en effet le contenu d'un fichier dans un tableau, chaque entrée étant une ligne du fichier.

file()

Retourne le contenu d'un fichier dans un tableau.

Syntaxe	`array file(string $nomFichier [, int $cheminInclude [, resource $contexte]])`
`$nomFichier`	Nom du fichier à lire. Cette fonction peut aussi lire les fichiers distants.
`$cheminInclude`	`TRUE` si le fichier doit être recherché dans les répertoires précisés dans `include_path`., `FALSE` sinon. La valeur par défaut est `FALSE`.
`$contexte`	Paramètre optionnel, utilisable depuis les versions 4.3 de PHP, spécifiant la ressource de contexte créée par `stream_context_create()` à utiliser. Voir le sous-chapitre sur les flux pour plus de renseignements sur les contextes de flux.
`retour`	Tableau indexé contenant chaque ligne du fichier, ou `FALSE` si une erreur est survenue.

Listing 9.5 : file.php

```
<?php
$buffer = file("message.txt");
for ($i=0;$i<count($buffer);$i++)
{
    echo "<font color='#cc0000'>message ".($i+1)."</font>:";
    echo $buffer[$i];
    echo "<br />";
}
?>
```

Si vous souhaitez simplement afficher le contenu du fichier sur la sortie, il existe une fonction plus pratique ; `readfile()` permet en effet ce type d'opération.

readfile()

Retourne, sur la sortie standard, le contenu du fichier passé en paramètre.

Syntaxe	`int readfile(string $nomFichier [, int $cheminInclude [, resource $contexte]]).`
`$nomFichier`	Nom du fichier à lire. Cette fonction peut aussi lire les fichiers distants.
`$cheminInclude`	`TRUE` si le fichier doit être recherché dans les répertoires précisés dans `include_path`., `FALSE` sinon. La valeur par défaut est `FALSE`.
`$contexte`	Paramètre optionnel, utilisable depuis les versions 4.3 de PHP, spécifiant la ressource de contexte créée par `stream_context_create()` à utiliser. Voir le sous-chapitre sur les flux pour plus de renseignements sur les contextes de flux.
`retour`	Retourne le nombre d'octets qui ont été lus, ou `FALSE` en cas d'erreur.

Listing 9.6 : readfile_01.php

```
<?php
$taille = readfile("http://www.mozilla.org");
echo "<hr />";
echo "Taille de la page = ".$taille." octets";
?>
```

Cette fonction est également pratique (associée à un en-tête approprié) pour forcer le téléchargement de fichiers, y compris s'il s'agit de fichiers habituellement interprétés par le navigateurs (*.txt*, *.html*, etc.).

Listing 9.7 : readfile_02.php

```
<?php
header("Content-type: application/octet-stream");
header("Content-disposition: attachment; filename=\"message.txt\");
readfile("message.txt");
?>
```

Le premier en-tête permet de "forcer" le type du document en un type théoriquement non interprété par le navigateur du client (sauf si ce dernier l'a configuré de manière inadéquate), ce qui aura pour effet de proposer la sauvegarde du fichier sur le disque. Le second en-tête permet de suggérer un nom de sauvegarde pour le fichier. Et, enfin, l'appel à `readfile()` envoie les données au client.

De la même façon, il est possible de rediriger un fichier sur la sortie standard à partir de la position courante du pointeur de lecture. `fpassthru()`, tout comme `readfile()`, est utilisé pour envoyer au client le contenu d'un fichier. Attention, `fpassthru()` ferme automatiquement le fichier (vous ne pouvez plus utiliser l'identifiant de fichier).

fpassthru()

Retourne sur la sortie standard le contenu d'un fichier à partir de la position courante du pointeur de lecture.

Syntaxe	`int fpassthru(resource $fp)`
`$fp`	Identifiant de fichier tel que retourné par `fopen()`, `popen()` et `fsockopen()`.
`retour`	Nombre d'octets renvoyés sur la sortie standard.

Listing 9.8 : fpassthru.php

```
<?php
//  Ouverture du fichier en mode lecture
$fp = fopen("message.txt","r");
//  Déplacement de la position courante de 20 octets.
fseek($fp, 20, SEEK_SET);
echo "<b>Affiche le fichier à partir du 20ème caractère.</b>";
echo "<br />";
fpassthru($fp);

echo "<hr />";
//  reouverture nécessaire du fichier en mode lecture
$fp = fopen("message.txt","r");
//  Déplacement de la position courante de 100 octets.
fseek($fp, 100, SEEK_SET);
echo "<b>Affiche le fichier à partir du 100ème caractère.</b>";
echo "<br />";
fpassthru($fp);
?>
```

Depuis PHP4.3, l'instruction `file_get_contents()` a fait son apparition. Pratique cette fonction permet de récupérer le contenu d'un fichier directement dans une variable chaîne.

file_get_contents()

Retourne le contenu d'un fichier dans une chaîne de caractères. Cette fonction est plus rapide à utiliser qu'une lecture en utilisant `fread()` ou `fgets()`.

Syntaxe :	`string file_get_contents(string $nomFichier [, bool $cheminInclude [, ressource $contexte]])`
`$nomFichier`	Nom du fichier à lire. Cette fonction peut aussi lire les fichiers distants.
`$cheminInclude`	`TRUE` si le fichier doit être recherché dans les répertoires précisés dans `include_path`., `FALSE` sinon. La valeur par défaut est `FALSE`.

`$contexte`	Paramètre optionnel spécifiant la ressource de contexte créée par `stream_context_create()` à utiliser. Voir le sous-chapitre sur les flux pour plus de renseignement sur les contextes de flux.
`retour`	Retourne dans une chaîne de caractères le contenu du fichier spécifié en argument. Si la lecture a rencontré un problème, la fonction retourne `FALSE`.

Voici une alternative à l'exemple vu avec `readfile()` :

```
<?php
$contenu = file_get_contents("http://www.mozilla.org");
echo $contenu ;
?>
```

Lecture et filtrage HTML

Le langage PHP possède une instruction permettant de lire une page HTML et de la retourner sans les différentes balises. Il s'agit de `fgetss()`, qui est similaire à la fonction `fgets()` – si ce n'est que les données retournées ne possèdent plus aucune balise HTML, hormis celles que le développeur aura demandé de conserver.

fgetss()

Lecture de la ligne courante du fichier sans les balises HTML le composant.

Syntaxe	`string fgetss(resource $fp, $nbOctet [, string $tagsAutorise])`
`$fp`	Identifiant sur un fichier ouvert à l'aide de l'instruction `fopen()`, `fsockopen()` ou `popen()`.
`$nbOctet`	Nombre d'octets qui doivent être lus.
`$tagsAutorise`	Les balises qui ne doivent pas être supprimées ; vous pouvez en spécifier plusieurs en les séparant par le caractère "|". Ex. : " |<b>| ".
`retour`	Chaîne contenant les données lues de la ligne courante sans les balises HTML. Retourne `FALSE` si le pointeur se trouve à la fin du fichier.

Listing 9.9 : fgetss.php

```
<?php
$fp = fopen("http://www.gnu.org/home.fr.html","r");
while ($ligne=fgetss($fp, 255, "<br>|<center>"))
{
    echo $ligne;
}
fclose($fp);
?>
```


REMARQUE

Les balises sur plusieurs lignes

Le fait que l'instruction `fgetss()` ne retourne qu'une ligne à la fois fait qu'il est impossible de supprimer les balises de deux lignes et plus. Avant toute chose, il est donc important de vérifier que le code HTML de votre fichier ne possède pas de balises sur plusieurs lignes. Si vous n'êtes pas l'auteur de la page que vous voulez traiter, commencez par mettre tout le code HTML sur une ligne unique, comme le fait le code ci-dessous.

```
<?php
$buffer = file("http://www.gnu.org/home.fr.html");
$fichier = join("",$buffer);
echo strip_tags($fichier,"<br>|<center>");
?>
```

La page retournée est ici complètement vide de balises HTML, à l'exception des balises `<br>` et `<center>` que nous avions choisi de conserver.

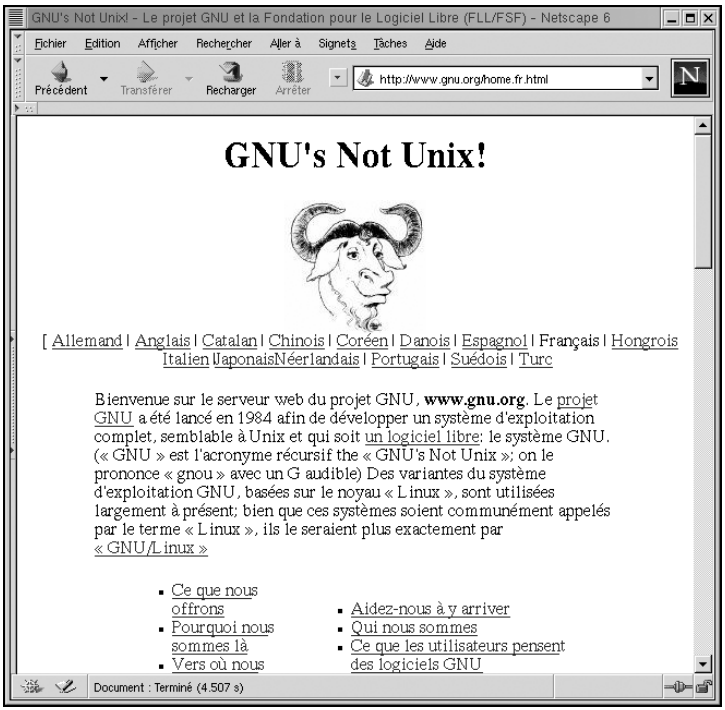


Figure 9.1 : *Le site original*

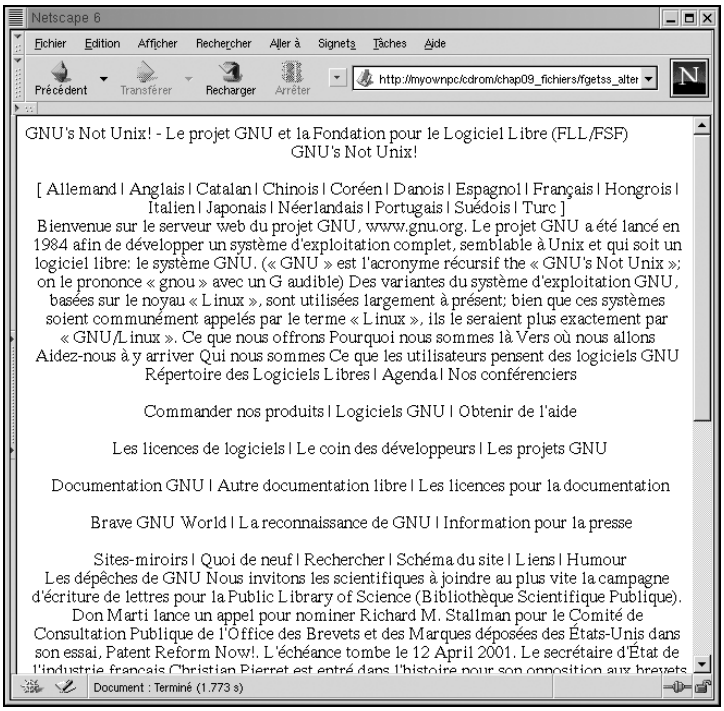


Figure 9.2 : *Le site après suppression des balises HTML*

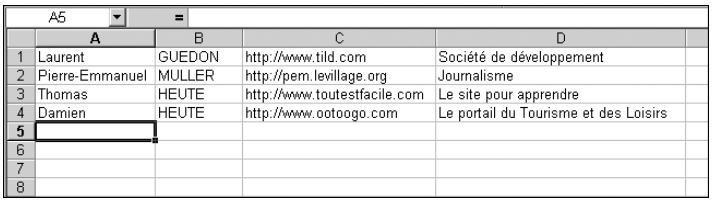
Lecture des fichiers CSV

Un fichier CSV (pour Comma Separated Value, valeurs séparées par des virgules) est un format utilisé par un grand nombre de tableurs et de bases de données pour l'importation et l'exportation des données. Ce format est constitué de lignes comportant différentes valeurs séparées par un délimiteur (à l'origine une virgule, mais généralement un point-virgule pour les versions françaises). Chacune des lignes du fichier représente alors une ligne d'un tableau, et chacune des valeurs, une donnée d'un champ de ce tableau. Voici un exemple de fichier CSV :

```
Laurent;GUEDON;http://www.tild.com;Société de développement
Pierre-Emmanuel;MULLER;http://pem.levillage.org;Journalisme
Thomas;HEUTE;http://www.toutestfacile.com;Le site pour apprendre
Damien;HEUTE;http://www.ootoogo.com;Le portail du Tourisme et des Loisirs
```

L'importation dans un tableau de type Excel ou Gnumeric donne un tableau de la forme ci-dessous (voir fig. 9.3) :

L'instruction `fgetcsv()` permet de lire aisément ce type de fichier. Nous allons nous en servir pour récupérer notre fichier de messages (il suffira d'indiquer que le délimiteur n'est pas une virgule mais une tabulation).



	A	B	C	D
1	Laurent	GUEDON	http://www.tild.com	Société de développement
2	Pierre-Emmanuel	MULLER	http://pem.levillage.org	Journalisme
3	Thomas	HEUTE	http://www.toutestfacile.com	Le site pour apprendre
4	Damien	HEUTE	http://www.ootoogo.com	Le portail du Tourisme et des Loisirs
5				
6				
7				
8				

Figure 9.3 : *Un tableau Excel après importation des données du fichier CSV*

fgetcsv()

Lit une ligne d'un fichier et retourne un tableau contenant les différents champs CSV.

Syntaxe	`array fgetcsv(resource $fp, int $nbOctet [, string $delimiteur])`
`$fp`	Identifiant sur un fichier ouvert à l'aide de l'instruction `fopen()`, `fsockopen()` ou `popen()`.
`$nbOctet`	Nombre d'octets qui doivent être lus au maximum.
`$delimiteur`	Caractère délimitant chacune des données (par défaut, le délimiteur est la virgule).
`retour`	Retourne un tableau contenant les différentes données d'une ligne du fichier CSV. Lorsque la fin du fichier est atteinte, la fonction retourne `FALSE`.

Et voilà ! Cette fonction est particulièrement bien adaptée à notre système de discussion.

Listing 9.10 : chat.php

```
<?php
//  Vérifie que les données sont bien postées
if($_POST["message"] && $_POST["pseudo"])
{
    //  Ouverture du fichier en écriture (mode ajout)
    $fp = fopen("message.txt", "a+");
    //  Récupération de l'heure
    $heure = date('H:m:s');
    //  Ecriture de l'heure et d'une tabulation
    fwrite($fp, $heure."\t");
    //  Ecriture du pseudo et d'une tabulation
    /*
        On utilise l'instruction htmlEntities afin de
        convertir les caractères spéciaux dans leur entité
        HTML
    */
    fwrite($fp, htmlEntities($_POST["pseudo"])."\t");
    //  Ecriture du message
    fwrite($fp, htmlEntities($_POST["message"])."\n");
    //  On ferme le fichier.
```

```
    fclose($fp);
}
?>

<html>
<title>Ecrire des messages dans un fichier</title>
</head>
<body>
<table border="0" width="100%">
<?php
//  Affiche les messages
$fp = fopen("message.txt","r");
while($donnee = fgetcsv($fp, 255, "\t"))
{
    echo "<tr>";
    echo "  <td><font color='#0000ff'>";
    echo $donnee[0];
    echo "  </font></td>";
    echo "  <td><b>";
    echo $donnee[1];
    echo "  </b></td>";
    echo "  <td>";
    echo $donnee[2];
    echo "  </td>";
    echo "</tr>";
}
fclose($fp);
?>
</table>
<hr />
<table border="0" width="100%">
    <tr>
    <form method="post">
    <td width="600">
        pseudo:
        <input type="text" name="pseudo"
               value="<?php htmlEntities($_POST["pseudo"]);?>"
               size="20" maxlength="20">
        message:
        <input type="text" name="message" size="40" maxlength="255">
        <input type="submit" name="envoyer" VALUE="&gt;&gt;">
    </td>
    </form>
    <td> </td>
    </tr>
</body>
</html>
```

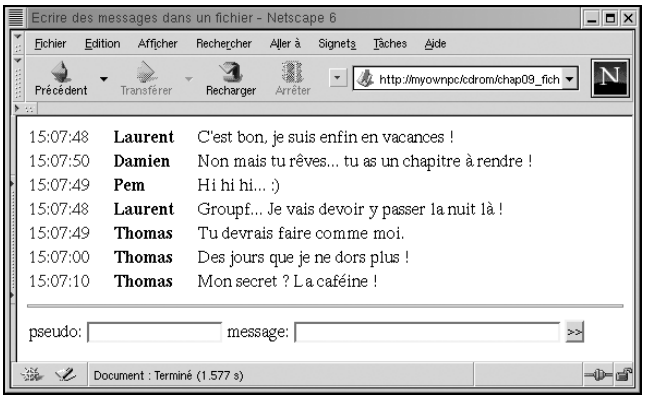


Figure 9.4 : *Voici le système de discussion qu'il ne vous reste plus qu'à perfectionner*

Lecture des fichiers .ini

L'installation d'un logiciel ou d'une application nécessite la création d'un certain nombre de paramètres dépendant de l'utilisateur ou de l'environnement. Il est courant de stocker ces informations dans un fichier du type *monFichier.ini*. Le fichier de configuration *php.ini* est un bon exemple de ce genre de configuration. Celui-ci contient alors, sous un format bien spécifique, différentes données, regroupées par section, qui peuvent être exploitées par la suite. Le fichier se présente de la façon suivante :

```
; Quelques lignes de commentaires.
; Le parseur ne les prendra pas en compte.

[section1]
utilisateur = 3
administrateur = 1
;rien = rien

[section2]
titre = Mon application
url = http://www.tild.com

[section3]
fichier = parse_ini_file.php
chemin = /home/e-smith/files/ibays/kangouroo/html/bible/fichier
droit = 755
```

Le langage PHP possède une fonction permettant de traiter les fichiers de ce type et d'en extraire les éléments qui le composent. L'instruction `parse_ini_file()` retourne les différentes données dans un tableau associatif. Vous pouvez récupérer un tableau sur deux niveaux comportant, dans le premier niveau, les différentes sections et, dans le second, les attributs et leurs valeurs.

parse_ini_file()

Retourne les différents éléments composant un fichier du type *.ini* et forme un tableau associatif.

Syntaxe	`array parse_ini_file(string $nomFichier [, boolean $traiteSection])`
`$nomFichier`	Nom du fichier à lire. Cette fonction peut aussi lire les fichiers distants en donnant une URL en paramètre.
`$traiteSection`	`TRUE` si vous souhaitez que les données de chaque section soient regroupées dans des tableaux, `FALSE` (par défaut) sinon.
`retour`	Selon les cas : Un tableau associatif ayant pour clés les noms des sections et pour valeur un tableau associatif ayant lui-même pour clé le nom du paramètre et pour valeur la valeur associée si `$traiteSection=TRUE`. Un tableau associatif ayant pour clés les noms des paramètres (toutes sections confondues) et pour valeur la valeur associée si `$traiteSection=FALSE`.

Listing 9.11 : parse_ini_file.php

```
<html>
<head>
    <title>Lecture d'un fichier CSV</title>
</head>
<body>
<table border="1" cellpadding="1" cellspacing="0">
    <tr>
        <td colspan="2">
<?php
echo "Affichage des données sans les différentes sections.";
echo "  </td>";
echo "</tr>";
$tableauIni = parse_ini_file("emma.ini");
while (list($key, $val) = each($tableauIni)) {
echo "<tr>";
echo "  <td>";
echo "$key";
echo "  </td>";
echo "  <td>";
echo "$val";
echo "  </td>";
echo "</tr>";
}
?>
</table>
<br />
<table border="1" cellpadding="1" cellspacing="0">
    <tr>
```

```
        <td colspan="2">
<?
echo "Affichage des données avec les différentes sections.";
echo "<br />";
echo "  </td>";
echo "</tr>";
$tableauIni = parse_ini_file("emma.ini", TRUE);
while (list($section, $tableauPar) = each($tableauIni)) {
    echo "<tr>";
    echo "  <td colspan='2'>";
    echo "  <b>".$section."</b>";
    echo "  </td>";
    echo "</tr>";
    while (list($key, $val) = each($tableauPar)) {
        echo "<tr>";
        echo "  <td>";
        echo "      $key";
        echo "  </td>";
        echo "  <td>";
        echo "      $val";
        echo "  </td>";
        echo "</tr>";
    }
}
?>
</table>
</body>
</html>
```

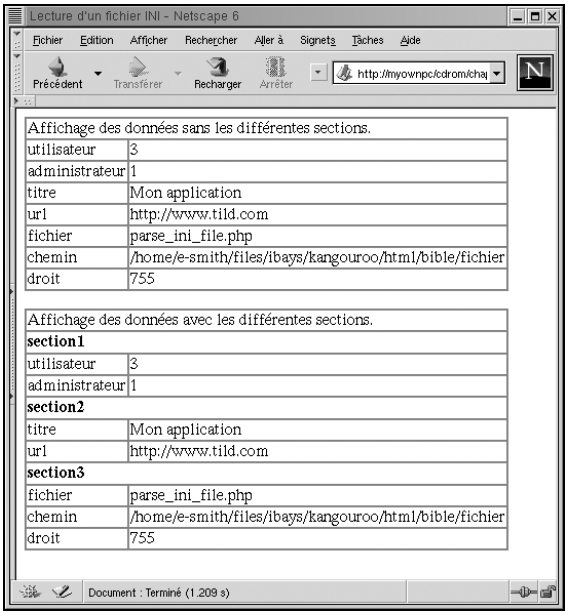


Figure 9.5 : *Le résultat de traitement du fichier emma.ini*

Positionner le pointeur de lecture/écriture

Lorsque l'on désire lire les données contenues dans un fichier, elles sont dans un premier temps copiées dans un espace mémoire (buffer). On dit que le mode de lecture est "bufferisé". Une fois le fichier ouvert, un pointeur de lecture se place automatiquement au début de ce fichier, chacun des appels à une instruction de lecture faisant avancer ce pointeur. L'avancée de ce pointeur peut être contrôlée par trois fonctions qui sont `feof()`, `fseek()` ou `rewind()`. La première instruction permet de déterminer si la fin du fichier ouvert est atteinte. La seconde instruction repositionne le pointeur de lecture (ou d'écriture) à une position spécifique. Quant à la dernière fonction, elle renvoie le pointeur au début du fichier.

feof()

Détermine si la fin du fichier est atteinte.

Syntaxe	`boolean feof(resource $fp)`
`$fp`	Identifiant sur un fichier ouvert à l'aide d'une instruction `fopen()`, `tmpfile()`, `fsockopen()` ou `popen()`.
`retour`	Retourne `TRUE` si le pointeur est à la fin du fichier, `FALSE` dans le cas contraire.

```
<?php
$fp = fopen("data.txt", "r");
//  Traitements divers
if (feof($fp))
{
    echo "Fin du fichier.<br />";
}
fclose($fp);
?>
```

Vous pouvez connaître la position courante du pointeur à l'aide la fonction `ftell()`.

ftell()

Retourne la position courante du fichier.

Syntaxe	`int ftell(resource $fp)`
`$fp`	Identifiant sur un fichier ouvert à l'aide d'une instruction `fopen()`, `tmpfile()`, `fsockopen()` ou `popen()`.
`retour`	Retourne la position courante du pointeur de lecture/écriture.

```
<?php
$fp = fopen("data.txt", "r");
```

```
echo "Position courante : ".ftell($fp);
fclose($fp);
?>
```

fseek()

Déplace le pointeur de lecture/écriture d'un fichier à une position spécifique.

Syntaxe	`int fseek(resource $fp, int $offset [, $origine])`
`$fp`	Identifiant sur un fichier ouvert à l'aide d'une instruction `fopen()`, `tmpfile()`, `fsockopen()` ou `popen()`.
`$offset`	Nombre indiquant le déplacement en octets (positif ou négatif) par rapport à l'origine sélectionnée.
`$origine`	Indique à partir de quel endroit doit être compté le déplacement. Au choix : `SEEK_SET` (= 0) à partir du début du fichier. `SEEK_CUR` (= 1) à partir de la position courante du pointeur (valeur par défaut). SEEK_END (= 2) à partir de la fin du fichier.
`retour`	`0` en cas de succès, et `-1` en cas d'erreur..

Il est à noter qu'un déplacement positif à partir de la fin du fichier n'entraîne pas d'erreur. En effet, déplacer un pointeur au-delà de EOF est possible.

Listing 9.12 : fseek.php

```
<?php
echo "taille : ".filesize("message.txt");
//  Affiche la taille du fichier en octets

echo "<br />";
$fp = fopen("message.txt","r");
fseek($fp, 20, SEEK_SET);
echo "Position par rapport au début du fichier (0+20) : ";
echo ftell($fp)."<br />";

fseek($fp, 20, SEEK_CUR);
echo "Position par rapport à la position courante (20+20) : ";
echo ftell($fp)."<br />";

fseek($fp, 20, SEEK_END);
echo "Position par rapport à la fin du fichier (Fin+20 ): ";
echo ftell($fp)."<br />";

fclose($fp);
?>
```

pourrait retourner un résultat du genre :

```
taille : 190
Position par rapport au début du fichier (0+20) : 20
Position par rapport à la position courante (20+20) : 40
Position par rapport à la fin du fichier (Fin+20 ): 210
```


ATTENTION

Le pointeur de lecture/écriture
Si vous ouvrez le fichier à l'aide de l'instruction `fopen()` *avec le mode* "`a`" *ou* "`a+`" *(ouverture pour ajout de données), le pointeur est systématiquement placé à la fin du fichier.*

rewind()

Repositionne le pointeur de lecture/écriture au début du fichier.

Syntaxe	`boolean rewind(resource $fp)`
`$fp`	Identifiant sur un fichier ouvert à l'aide d'une instruction `fopen()`, `tmpfile()`, `fsockopen()` ou `popen()`.
`retour`	`TRUE` si l'opération s'est déroulée sans problème, et `FALSE` si le pointeur n'a pas pu être repositionné.

```
<?php
$fp = fopen("message.txt", "r");
//  Traitements divers
if (rewind($fp))
{
    echo "Le pointeur du fichier a été repositionné au début";
    echo "<br />";
}
fclose($fp);
?>
```

Tronquer un fichier

Une fonction particulière permet de tronquer les fichiers qui ont été ouverts en mode écriture. Le langage PHP permet de récupérer une partie des informations contenues dans un fichier et de les réécrire en supprimant le reste des données. L'instruction `ftruncate()` ne fonctionne que dans le cas où le fichier a été ouvert en écriture, c'est-à-dire les modes écriture seule ('`w`', '`a`'), lecture et écriture ('`w+`', '`a+`').

ftruncate()

Tronque un fichier (i.e. ne conserve que les premiers octets du fichier).

Syntaxe	`int ftruncate(resource $fp, int $taille)`
`$fp`	Identifiant sur un fichier ouvert à l'aide d'une instruction `fopen()`, `tmpfile()`, `fsockopen()` ou `popen()`.
`$taille`	Nouvelle taille du fichier.
`retour`	`1` (mais pas strictement `TRUE`) si le fichier a bien été tronqué, `0` (mais pas strictement `FALSE`) sinon.

Imaginons le contenu d'un fichier *dedicace.txt*.

```
Je dédicace ce livre à tous mes amis
Et plus particulièrement à Myriam si adorable.
A Amiel pour son soutien et ses compétences.
A Damien pour savoir botter le cul quand il le faut.
A Thomas pour son calme à faire passer un bouddha pour un rocker sous ecsta.
```

Ce fichier est bien trop long et nous ne voulons conserver que les deux premières lignes. La principale difficulté consiste donc à déterminer la nouvelle taille que nous souhaitons donner à ce fichier. Pour cela, il suffit de lire les deux premières lignes, et de déterminer quelle est la position atteinte par le pointeur de fichier (grâce à `ftell()`). Cette position correspondra à la taille voulue.

Listing 9.13 : ftruncate.php

```
<?php
//  Ouvre un fichier texte en mode lecture et écriture
if ($fp = fopen("dedicace.txt", "r+"))
{
    //  Lecture des deux premières lignes
    fgets($fp, 255);
    fgets($fp, 255);
    //  On tronque à une taille égale
    //  à la position courante du pointeur de
    //  lecture
    ftruncate($fp, ftell($fp));
    //  Fermeture du fichier
    fclose($fp);
    //  Affichage du fichier une fois tronqué
    readfile("dedicace.txt");
} else {
    echo "Impossible d'ouvrir le fichier dédicace.";
}
?>
```

Ce qui donnera le résultat suivant :

```
Je dédicace ce livre à tous mes amis
Et plus particulièrement à Myriam si adorable.
```

Gestion de l'espace tampon

L'écriture dans un fichier est effectuée en deux temps. Les données sont d'abord copiées vers un buffer (espace mémoire) et sont ensuite écrites sur le fichier ouvert. Ce système de bufferisation permet de gagner du temps, car les appels système se font plus rares. Imaginons que l'on désire écrire dans un fichier 20 lignes de 200 caractères. Il faudra alors normalement 20 appels système pour écrire la totalité des données dans le fichier. L'utilisation du buffer permet d'écrire une seule fois les 4 000 (20*200) caractères, c'est-à-dire que le programme ne fait appel qu'une fois au système d'exploitation, et ne sollicite qu'une fois le disque dur pour écrire dans le fichier. Le langage PHP possède, par défaut, un buffer de 8 Ko, mais il est possible de lui fixer une autre taille avec l'instruction `set_file_buffer()`. Il peut être particulièrement intéressant d'augmenter cette valeur si votre script doit traiter des fichiers de très grande taille. Depuis PHP 4.3, l'instruction `stream_set_write_buffer()` remplace cette fonction. `set_file_buffer()` demeure un alias pour des raisons de retro-compatibilités.

stream_set_write_buffer()

Fixe la taille du buffer.

Syntaxe	`int stream_set_write_buffer(ressource $fp, int $taille)`
`$fp`	Identifiant sur un fichier ouvert à l'aide d'une instruction `fopen()`, `tmpfile()`, `fsockopen()` ou `popen()`.
`$taille`	Nouvelle taille du buffer.
`retour`	`0` si l'instruction se déroule convenablement, `EOF` dans le cas contraire.

Du fait de la présence de ce système de cache, les données écrites dans un fichier ne sont pas toujours disponibles immédiatement (pour les scripts cherchant à en lire le contenu). Les données ne seront effectivement accessibles qu'une fois le buffer vidé (ce qui intervient automatiquement dès qu'il est plein ou manuellement lorsque vous appelez la fonction `fflush()`).

Notez également que le buffer est automatiquement vidé lors de l'appel à `fclose()`. A contrario, si vous quittez l'exécution de votre programme sans faire appel à `fclose()`, vous courez le risque de perdre les données restant dans le cache.

fflush()

Écrit et vide le buffer d'un fichier.

Syntaxe	`boolean fflush (resource $fp)`

`$fp`	Identifiant sur un fichier ouvert à l'aide d'une instruction `fopen()`, `tmpfile()`, `fsockopen()` ou `popen()`.
`retour`	Retourne `TRUE` si le buffer a été vidé dans le fichier, ou `FALSE` dans le cas contraire.

Listing 9.14 : fflush.php

```
<?php
$fp = fopen("twiki.txt", "w");
if ($fp) {
    // Les données seront écrites tous les 500 octets
    set_file_buffer($fp, 500);

    // Cette phrase fait moins de 500 octets
    // elle n'est donc pas immédiatement écrite
    fputs($fp, "Bidibidibidi comme dirait Twiki.");

    // Mais qu'à cela ne tienne. Nous allons forcer l'écriture
    fflush($fp);

    // Cette phrase attendra fclose() pour être écrite
    fputs($fp, "Glop Glop comme dirait Pifou.");

    fclose($fp);
}
?>
```

Lister le contenu d'un dossier

Nous avons vu que, pour visualiser le contenu d'un fichier, il suffit de l'ouvrir et d'obtenir ainsi un pointeur vers ce fichier. L'ouverture d'un répertoire est similaire à celle d'un fichier. Vous créez, à l'aide de la fonction `openDir()`, un identifiant, qui vous permet par la suite de lire ce dossier à la façon d'un fichier.

openDir()

Ouvre le répertoire désigné et retourne un pointeur dessus.

Syntaxe	`resource openDir(string $chemin)`
`$chemin`	Chemin du dossier à ouvrir.
`retour`	Pointeur sur le dossier ouvert.

Ne pas oublier de fermer le dossier à l'aide de l'instruction `closeDir()`.

closeDir()

Fermer un dossier préalablement ouvert avec la fonction `openDir()`.

Syntaxe	`void closeDir(resource $repertoire)`
`$repertoire`	Pointeur sur un dossier ouvert avec l'instruction `openDir()`.

```
<?php
//  Ouverture du dossier parent "/monrepertoire"
$repertoire = openDir("/monrepertoire");
//  Placer les différents traitements à effectuer
//  ...
//  Fermeture du dossier
closeDir($repertoire);
?>
```

Une fois un pointeur obtenu sur le répertoire, il suffit, pour lister le contenu d'un dossier, de faire appel à la fonction `readDir()`. Le pointeur de lecture se place au début du répertoire et chacun des appels à la fonction `readDir()` le fait avancer sur l'entrée suivante, à savoir le nom du fichier ou répertoire suivant. De proche en proche, il est ainsi possible de lire la totalité du contenu d'un dossier.

readDir()

Lecture des entrées dans un dossier.

Syntaxe	`string readDir(resource $repertoire)`
`$repertoire`	Pointeur sur un dossier ouvert avec l'instruction `openDir()`.
`retour`	Retourne le nom du fichier désigné par le pointeur de lecture.

Ainsi, pour afficher le contenu d'un répertoire, nous pouvons utiliser la fonction suivante :

Listing 9.15 : readdir.php

```
<?php
// La fonction d'exploration
function explorer($chemin) {
    $repertoire = openDir($chemin);
    while ($fichier = readDir($repertoire)) {
        //  Inutile d'afficher les entrées . et ..
        if (($fichier != ".")&&($fichier != "..")) {
            // n'oublions pas d'ajouter
            // le chemin au nom du fichier
            echo $chemin."/".$fichier<br />";
            }
        }
```

```
    // C'est fini. On ferme !
    closeDir($repertoire);
}

// Définition du chemin à explorer
$cheminRep = ".";

// L'appel à la fonction
explorer($cheminRep);
?>
```

qui pourrait donner :

```
./fichier0.php
./fichier1.php
./fichier2.php
./repertoire
```

Nous pouvons, de plus, imaginer que tous les sous-répertoires soient listés, permettant ainsi d'avoir la liste complète des fichiers contenus dans le dossier courant. Pour cela, nous allons utiliser les propriétés de récursivité du langage PHP. Nous devrons en outre être capables de distinguer les répertoires des fichiers. Pour cela, nous devrons faire appel à la fonction `is_dir()` qui indique s'il s'agit d'un répertoire. Et, enfin, afin de distinguer le parcours du répertoire du traitement du contenu, nous retournerons un tableau contenant les noms des fichiers.

Listing 9.16 : readdir_recursif.php

```
<?php
// La fonction d'exploration
function explorer($chemin, $recursif=FALSE) {
    $listeFichier = array();

    $repertoire = openDir($chemin);
    while ($fichier = readDir($repertoire)) {
        // Inutile de tenir compte des entrées . et ..
        if (($fichier != ".")&&($fichier != "..")) {
            // Est-ce que $file est un répertoire ?
            // Pour le savoir il suffit d'appeler is_dir
            // mais attention n'oublions pas d'ajouter
            // le chemin au nom du fichier
            if (is_dir($chemin."/".$fichier)&&($recursif)) {
                // oui ? alors explorons-le
                // et ajoutons le résultat à la liste de fichiers
                $listeFichier = array_merge($listeFichier,
                                            explorer($chemin."/".$fichier,
                                                     $recursif));
            } else {
                // sinon, c'est un fichier et on l'ajoute
                // à la liste des fichiers
                $listeFichier[] = $chemin."/".$fichier;
            }
```

```
        }
    }
    // C'est fini. On ferme !
    closeDir($repertoire);
    // et on retourne le résultat
    return $listeFichier;
}

// Définition du chemin à explorer
$cheminRep = "../.";

// L'appel à la fonction
$fichiers = explorer($cheminRep, TRUE);

// Affichage (à titre démonstratif)
for ($i=0; $i<count($fichiers); $i++) echo $fichiers[$i]."<br />";
?>
```

Ce script aisément réutilisable retournera donc quelque chose comme :

```
./fichier0.php
./fichier1.php
./fichier2.php
./repertoire/fichier0.php
./repertoire/fichier1.php
./repertoire/fichier2.php
./repertoire/fichier3.php
./repertoire/fichier4.php
```

Dans certains cas, comme la galerie d'images (un grand classique), il faut filtrer les noms des fichiers (généralement d'après leur extension). Nous allons donc perfectionner encore notre fonction pour préciser une expression régulière à laquelle doit répondre le nom du fichier. Une fois la liste des fichiers récupérée, il suffira d'afficher les images.

Listing 9.17 : readdir_filtre.php

```
<?php
// La fonction d'exploration
// $chemin : Répertoire à explorer
// $recursif : TRUE si l'exploration doit être récursive
// $filtre : Expression régulière de filtrage des fichiers

function explorer($chemin, $recursif=FALSE, $filtre=NULL) {
    $listeFichier = array();

    $repertoire = openDir($chemin);
    while ($fichier = readDir($repertoire)) {
        // Inutile de tenir compte des entrées . et ..
        if (($fichier != ".")&&($fichier != "..")) {
            // Est-ce que $file est un répertoire ?
            // Pour le savoir il suffit d'appeler is_dir
            // mais attention n'oublions pas d'ajouter
```

```
            // le chemin au nom du fichier
            if (is_dir($chemin."/".$fichier)&&($recursif)) {
                // oui ? alors explorons-le
                // et ajoutons le résultat à la liste de fichiers
                $listeFichier = array_merge($listeFichier,
                                    explorer($chemin."/".$fichier,
                                    $recursif, $filtre));
            } else {
                // sinon, c'est un fichier et s'il répond
                // aux critères de l'expression régulière
                // on l'ajoute a la liste des fichiers
                if (is_null($filtre)||eregi($filtre, $fichier)) {
                    $listeFichier[] = $chemin."/".$fichier;
                }
            }
        }
    }
    // C'est fini. On ferme !
    closeDir($repertoire);
    // et on retourne le résultat
    return $listeFichier;
}

// Définition du chemin à explorer
$cheminRep = "../.";

// L'appel à la fonction
$fichiers = explorer($cheminRep, TRUE, ".gif|.jpg|.png");

// Affichage (à titre démonstratif)
echo "<html><body>";
echo "<table width='100%' border='0'>";
for ($i=0; $i<count($fichiers); $i++) {
    // 5 images par ligne
    if ($i%5 == 0) echo "<tr></td>"; else echo "<td>";
    echo "<img src='".$fichiers[$i]."'><br />";
    if (($i+1)%5 == 0) echo "</td></tr>\n"; else echo "</td>";
}
echo "</table>";
echo "</body></html>";
?>
```

Figure 9.6 : *Notre galerie d'images*

L'ordre des fichiers
Vous pouvez observer que la liste des fichiers n'est pas donnée dans l'ordre alphabétique. Si vous voulez ordonner cette liste, il vous suffit d'appliquer la fonction `sort()` *au tableau.*

Dans les exemples précédents, nous avons opté pour une solution retournant la liste des fichiers dans un tableau. Cette méthode peut nécessiter beaucoup de mémoire si la liste est longue. Une autre solution aurait pu consister à ajouter à la fonction un paramètre permettant de préciser le nom d'une fonction à appeler dès qu'un nouveau fichier est rencontré.

Pointeur de lecture du répertoire

Lors de la lecture du contenu du dossier, vous pouvez déplacer le pointeur à la première entrée à l'aide de l'instruction `rewindDir()`.

rewindDir()

Déplace le pointeur de lecture du dossier à la première entrée.

Syntaxe	`void rewindDir(resource $repertoire)`
`$repertoire`	Pointeur sur un dossier ouvert avec l'instruction `opendir()`.

```
<?php
$repertoire = openDir("./");
while ($fichier = readDir($repertoire)) {
    echo $fichier."<br />";
}
//  Affichage de la liste des fichiers du dossier.

while ($fichier = readDir($repertoire)) {
    echo $fichier."<br />";
}
//  Le programme n'affiche rien.

//  On déplace le pointeur sur la première entrée
rewindDir($repertoire);

while ($fichier = readDir($repertoire)) {
    echo $fichier."<br />";
}
//  Affichage de la liste des fichiers du dossier.
closeDir($repertoire);
?>
```

Rechercher un fichier dans un répertoire

Depuis la version 4.3 du langage PHP il est possible d'effectuer directement des recherches de fichier à l'aide de la fonction `glob()`. Ainsi, la fonction `openDir()` peut être avantageusement remplacée par `glob()` lorsqu'il est nécessaire de lister le contenu d'un répertoire en utilisant un filtre (masque).

glob()

Retourne les fichiers vérifiant un masque.

Syntaxe :	`array glob(string $masque [, int $param])`
`$masque`	Masque permettant de filtrer les fichiers à retourner.
`$param`	Option affectant le résultat : `GLOB_MARK` : Ajoute un slash (caractère /) à la fin de chaque répertoire. `GLOB_NOSORT` : Indique de ne pas utiliser l'ordre alphabétique pour retourner les résultats.

	GLOB_NOCHECK : Retourne le masque si aucun fichier ne correspond au masque $masque. GLOB_NOESCAPE : Enlève les anti-slash devant les méta-caractères comme l'espace. GLOB_BRACE : utilise la formulation {a, b, c} pour rechercher dans le masque les chaînes a, b ou c. GLOB_ONLYDIR : Ne retrourne que les répertoires.
retour	Retourne un tableau contenant la liste des résultats. En cas d'erreur, la fonction retourne FALSE.

```
<?php
/*
    Répertoire contenant les fichiers emma.txt, damien.txt,
    laurent.txt, thomas.txt, pem.txt
    et les répertoire bible et php.
*/
    print_r(glob("p*"));

    print_r(glob("p*", GLOB_MARK));

    print_r(glob("p*", GLOB_NOSORT));

    print_r(glob("*nux", GLOB_NOCHECK));

    print_r(glob("{la,da,th}*.txt", GLOB_BRACE));

    print_r(glob("*", GLOB_ONLYDIR));
?>
```

```
Array
(
    [0] => pem.txt
    [1] => php
)
Array
(
    [0] => pem.txt
    [1] => php/
)
Array
(
    [0] => php
    [1] => pem.txt
)
Array
(
    [0] => *nux
)
Array
(
    [0] => laurent.txt
    [1] => damien.txt
```

```
    [2] => thomas.txt
)
Array
(
    [0] => bible
    [1] => php
)
```

Manipulation de fichiers et répertoires

Maintenant que nous avons une liste de fichiers, nous pouvons nous intéresser à la manipulation de ces fichiers. Le langage PHP possède une liste de fonctions qui vont nous permettre de copier, renommer, supprimer ou même créer des liens symboliques sur les fichiers.

Dans un premier temps, intéressons-nous à la copie de fichier. L'instruction `copy()` permet à votre programme d'accéder au système de fichier, et de copier le fichier désigné vers le dossier et sous le nom que vous aurez décidé.

copy()

Copie un fichier.

Syntaxe	`boolean copy(string $source, string $destination)`
`$source`	Chemin vers le fichier source à copier.
`$destination`	Chemin de destination du fichier.
`retour`	Retourne `TRUE` si le fichier a bien été copié, `FALSE` sinon.

```
<?php
copy("emma.jpg", "../emma.jpg");
//  Cet exemple copie le fichier emma.jpg
//  vers le dossier en dessous
?>
```

Le langage PHP possède deux instructions `link()` et `symlink()` qui permettent respectivement de créer un lien dur ou un lien symbolique sur un fichier.

link() (non disponible sous Windows)

Créer un lien dur sur un fichier.

Syntaxe	`boolean link(string $nomFichier, string $nomLien)`

`$nomFichier`	Le nom du fichier sur lequel le lien doit être créé.
`$nomLien`	Le nom du lien à créer.
`retour`	`TRUE` si le lien a bien été créé, `FALSE` dans le cas contraire (avec un message de type "warning" sous Windows).

```
<?php
if (link("/home/laurent/emma.png","/home/damien/emma.png") {
    echo "Le lien a bien été créé.";
} else {
    echo "Le lien n'a pas été créé.";
}
?>
```

symLink() (non disponible sous Windows)

Créer un lien symbolique sur un fichier.

Syntaxe	`boolean symLink(string $nomFichier, string $nomLienSymb)`
`$nomFichier`	Le nom du fichier sur lequel le lien doit être créé.
`$nomLienSymb`	Le nom du lien symbolique à créer.
`retour`	`TRUE` si la fonction n'a pas rencontré de problème, `FALSE` sinon (avec un message de type "warning" sous Windows).

```
<?php
if (symlink("/home/laurent/emma.png","/home/damien/emma.png") {
    echo "Le lien symbolique a bien été créé.";
} else {
    echo "Le lien symbolique n'a pas été créé.";
}
?>
```

Les programmes exploitant les fonctions de manipulation de fichiers doivent souvent utiliser des fichiers temporaires. Il est en effet souvent intéressant de créer des fichiers tampons afin de stocker les données temporairement. La fonction `tempNam()` permet cette manipulation en créant, sur un répertoire, un fichier possédant un nom unique.

tempNam()

Créer un fichier temporaire dans un répertoire fourni en paramètre.

Syntaxe	`string tempNam(string $dossier, string $prefixe)`
`$dossier`	Dossier où doit être créé le fichier temporaire. Si le nom du dossier est mis à `NULL`, le fichier sera créé dans le répertoire temporaire du système

	(dépend de la configuration mais, généralement, /tmp sous Linux et *C:\windows\temp* sous Windows).
`$prefixe`	Préfixe du nom de fichier à créer (permet de distinguer plus simplement les fichiers temporaires des autres). Si vous ne souhaitez pas en préciser, mettez cette valeur à `NULL`.
`retour`	Le nom du fichier qui a été créé. Si une erreur est survenue, `FALSE` sera renvoyé.

```
<?php
$nomTemporaire = tempNam(NULL, "php_");
echo "Le fichier ".$nomTemporaire." vient d'être créé.";
?>
```

La manipulation de fichiers suppose aussi la possibilité de supprimer. L'instruction `unlink()` efface définitivement un fichier (ou un lien) du disque.

unlink()

Destruction d'un fichier ou d'un lien sur un fichier (pour les répertoires voir `rmdir()`).

Syntaxe	`boolean unlink(string $nomFichier)`
`$nomFichier`	Nom et chemin d'accès du fichier à supprimer.
`retour`	`TRUE` si le fichier a bien été supprimé, `FALSE` dans le cas contraire.

```
<?php
if (unlink("/home/laurent/emma.jpg") {
    echo "Le fichier a été effacé";
} else {
    echo "Le fichier n'a pas été détruit.";
}
?>
```

Nous avons vu comment copier un fichier et le supprimer ; ces deux instructions que sont `copy()` et `unlink()` seraient amplement suffisantes pour nous permettre de déplacer un fichier d'un endroit à un autre si le langage PHP ne possédait pas l'instruction `rename()`. Ainsi, en une seule commande, il est possible de déplacer le fichier et (ou) de le renommer.

rename()

Renomme et (ou) déplace un fichier ou un répertoire.

Syntaxe	`boolean rename(string $ancienNom, string $nouveauNom)`

`$ancienNom`	Ancien nom ou ancien chemin du fichier.
`$nouveauNom`	Nouveau nom ou nouvel emplacement du fichier.
`retour`	Si le changement de nom s'est bien déroulé, l'instruction retourne `TRUE`, `FALSE` sinon.

```
<?php
if (rename("/home/laurent/emma.png", "/home/thomas/emma.png")) {
    echo "Le fichier a été déplacé avec succès.";
} else {
    echo "Impossible de déplacer le fichier.";
}
?>
```

rmDir()

Supprime un répertoire à la condition que celui-ci soit vide.

Syntaxe	`boolean rmDir(string $nomDossier)`
`$nomDossier`	Chemin d'accès du dossier à supprimer.
`retour`	Retourne `TRUE` si le dossier a bien été supprimé, `FALSE` sinon.

```
<?php
if (rmdir("/home/laurent/emma") {
    echo "Le dossier a été effacé";
} else {
    echo "Le dossier n'a pas été détruit.";
}
?>
```

Le langage PHP permet, bien évidemment, de créer des dossiers. Pour cela, vous disposez de l'instruction `mkDir()`. Vous pourrez alors préciser les droits sur le répertoire ainsi créé. Bien entendu, le mode ne concerne que le système UNIX et non les plateformes Windows.

mkDir()

Crée un dossier et lui affecte un mode.

Syntaxe	`boolean mkDir(string $nomDossier [, int $mode])`
`$nomDossier`	Nom du dossier à créer.
`$mode`	Droits réclamés pour le dossier à créer (inopérant sous Windows).
`retour`	Retourne `TRUE` si le dossier a bien été créé ; à l'inverse, notamment si le dossier existe déjà, l'instruction renvoie `FALSE`.

```
<?php
if (mkDir("nouveaurepertoire",0755) {
    echo "Le dossier a été créé.";
} else {
    echo "Le dossier n'a pas été créé.";
}
?>
```

Il est probable que le dossier que vous créerez ne possédera pas exactement les droits que vous aurez spécifiés. En effet, chaque utilisateur (y compris le compte sous lequel tourne le serveur web) possède dans son environnement un paramètre appelé *umask*. Ce paramètre définit les droits d'accès qui sont, par défaut, retirés lors de la création des fichiers et répertoires.

Concernant les fichiers, les droits demandés par défaut sont 0666. Ainsi, un utilisateur qui possède un umask en 0000 créera par défaut des fichiers ayant les droits 0666 (0666 - 0000 en octal), alors qu'un utilisateur avec un umask en 0022, créera des fichiers ayant les droits 0644 (0666 - 0022 en octal).

Le principe est le même pour les répertoires. Si vous spécifiez des droits 0777, alors l'umask sera appliqué. S'il est à 0022, les droits effectifs du répertoire seront 0755 (0777 - 0022 en octal).

Si vous voulez créer des dossiers dans un mode particulier, vous serez peut-être amené à utiliser la fonction `umask()` qui permet de modifier l'umask pendant la durée de l'exécution du script (si PHP est compilé en module ou, au-delà, s'il est exécuté en CGI).

RENVOI

Pour plus d'informations sur la compilation de PHP, reportez-vous au chapitre "Prise en main".

umask() (inopérant sous Windows)

Change (ou retourne) la valeur de l'umask.

Syntaxe	`int umask([int $nouvMask])`
`$nouvMask`	Nouvel umask.
`retour`	Retourne la valeur précédente de l'umask (`0` sous Windows).

```
<?php
//  Modification de l'umask
$ancienUmask = umask(0022);
if (mkdir("/home/laurent/emma",0777) {
    echo "Le dossier possède le mode 755.";
} else {
    echo "Le dossier n'a pas été créé.";
}
//  réinitailisation de l'umask dans sa valeur initiale.
umask($ancienUmask);
?>
```

Modification des permissions

Lorsque vous désirez installer un logiciel, il est probable que, dans les notes d'installation, l'on vous demande de modifier les permissions de certains répertoires et fichiers en 777 ; cela pour la raison suivante :

généralement, vous placez les fichiers sur le serveur à l'aide d'un client FTP. Ceux-ci appartiennent alors au compte et au groupe utilisés lors de l'accès FTP. Les permissions sont donc limitées à l'utilisateur du compte FTP. De ce fait, le serveur HTTP (qui tourne en général sous un autre utilisateur, qui peut être, par exemple, `apache` ou `nobody`) n'a pas les droits d'accès sur ces fichiers. Le serveur ne peut alors pas écrire dans les répertoires ou les fichiers. La solution de facilité est donc de modifier les permissions sur les fichiers que le serveur est censé retoucher en donnant les droits en lecture, écriture et exécution à tous (0777).

Cela est aussi valable dans le sens inverse. Le serveur web peut créer des fichiers que l'utilisateur ne peut ensuite modifier, car il ne possède pas les droits d'écriture sur ledit fichier.

Le langage PHP permet la modification des propriétaires des fichiers et des répertoires, ainsi que la gestion de leurs droits. Il s'agit tout simplement des équivalents des commandes `chmod`, `chown` et `chgrp` disponibles sous Linux.

Attention, la modification des droits sur les fichiers suppose que le serveur HTTP possède les permissions nécessaires à cette modification. En règle générale, l'utilisateur qui effectue les changements est l'utilisateur propriétaire des fichiers ou fait partie du groupe propriétaire.

chmod() (inopérant sous Windows)

Modifie les permissions de lecture, d'écriture et d'exécution d'un fichier.

Syntaxe	`boolean chmod(string $nomFichier, int $mode)`
`$nomFichier`	Chemin d'accès au fichier à modifier.
`$mode`	Indique en octal le nouveau mode (ex. : 0755).
`retour`	`TRUE` si le changement de mode se déroule sans problème (et sous Windows), `FALSE` sinon.

Il est, là aussi, conseillé de placer un zéro devant le mode afin de préciser que celui-ci est donné en octal et non en décimal.

```
<?php
if (chmod("emma.txt", 0755)) {
    echo "Le changement de permission est un succès.";
} else {
    echo "Echec dans le changement de permission.";
}
?>
```

chown() (inopérant sous Windows)

Change le propriétaire du fichier.

Syntaxe	`boolean chown(string $nomFichier, mixed $utilisateur)`
`$nomFichier`	Nom du fichier à modifier.
`$utilisateur`	L'utilisateur à qui l'on veut donner les droits. Peut être précisé soit par son UID (identifiant d'utilisateur) soit par son login (nom d'utilisateur).
`retour`	`TRUE` si la modification a bien été effectuée (ou sous Windows), `FALSE` dans le cas contraire.

```
<?php
if (chown("fichier.txt", "damien")) {
    echo "Le changement d'utilisateur est un succès.";
} else {
    echo "Echec dans le changement d'utilisateur.";
}
?>
```

chgrp() (inopérant sous Windows)

Change le groupe propriétaire du fichier.

Syntaxe	`boolean chgrp(string $nomFichier, mixed $groupe)`
`$nomFichier`	Nom du fichier à modifier.
`$groupe`	Le groupe à qui l'on veut donner les droits. Peut être précisé soit par son GID (identifiant de groupe) soit par son nom.
`retour`	`TRUE` si la modification du groupe a été effectuée avec succès, `FALSE` dans le cas contraire (et sous Windows).

```
<?php
if (chgrp("fichier.txt", "invite")) {
    echo "Le changement de groupe est un succès.";
} else {
    echo "Echec dans le changement du groupe.";
}
?>
```

Upload de fichiers

Le terme upload désigne l'action de transférer, vers un serveur distant, un fichier qui se situe en local (sur le poste du client). L'upload de fichiers à l'aide d'un formulaire HTML n'est possible que depuis la version 1.1 de la norme HTTP. Le navigateur peut, à l'aide d'une méthode `POST`,

expédier un fichier sur le serveur dans un répertoire temporaire, qu'il soit d'un format texte ou binaire. Le code HTML permettant cette action est de la forme :

Listing 9.18 : upload_form.html

```
<html>
<head>
    <title>Upload de fichiers</title>
</head>
<body>
    <form action="upload.php" enctype="multipart/form-data" method="post">
    <input type="hidden" name="MAX_FILE_SIZE" value="1024" />
    Uploader le fichier : <input name="fichier" type="file" /><br />
    <input type="submit" value="Go &gt;&gt;">
    </form>
</body>
</html>
```

Ce formulaire contient deux balises particulièrement importantes :

- La balise `<form>` avec l'attribut `enctype="multipart/form-data"` indique au navigateur la façon dont les données seront envoyées (donc ici, sous forme de données brutes), alors que la valeur par défaut est `application/x-www-form-urlencoded`. L'envoi de fichiers ne peut se faire qu'avec la méthode `POST`, d'où la présence de l'attribut `method="post"`.
- La balise `<input>` avec l'attribut `type="file"`. Cette balise se traduit par la présence, sur le formulaire, d'un bouton permettant la sélection du fichier (sur le poste client).

Il est également conseillé de spécifier une balise supplémentaire contenant un champ caché qui précise la taille maximale (en octets) des fichiers à transférer. `<input type="hidden" name="MAX_FILE_SIZE" value="1024" />` (ici, pour des fichiers de 1 Ko au maximum).


ATTENTION

Il y a toujours un risque de casse
Prenez garde : le nom du champ `MAX_FILE_SIZE` doit obligatoirement être en majuscules.

Lorsqu'un tel formulaire est "soumis", le client envoie plusieurs lignes supplémentaires dans l'en-tête HTTP. Habituellement, pour des données classiques, c'est-à-dire qui ne sont pas des upload de fichier, le navigateur ajoute une ligne `Content-Disposition: form-data; name="leNomDuChamp"` suivie d'un retour chariot et de la valeur du champ. Pour chaque champ de type fichier, le navigateur ajoute une ligne d'en-tête `Content-Disposition: form-data; name="leNomDuChamp"; filename="monFichier.gif"` suivie d'un retour chariot, d'une ligne décrivant le contenu du fichier `Content-Type: image/gif` et, à nouveau, d'un retour chariot et, enfin, du contenu du fichier à envoyer sur le serveur.

Au final, le fichier est réceptionné dans un répertoire temporaire du serveur. Notez que le nom du fichier dans ce répertoire est indépendant du nom du fichier que l'utilisateur a transmis. En

effet, le fichier porte lui-même un nom temporaire. Il vous faudra alors renommer et déplacer le fichier dans un lieu sûr (à moins que vous ne souhaitiez seulement le consulter, et non pas le conserver).

Taille maximale du fichier

L'ajout du champ caché `MAX_FILE_SIZE` *n'est en rien une garantie que des fichiers de plus petite taille puissent être envoyés sur le serveur. En effet, la taille des fichiers uploadés est limitée par différents autres paramètres. La limite peut être imposée par PHP via l'option* `upload_max_filesize` *(par défaut 2 Mo) du fichier php.ini et, éventuellement, par le serveur web lui-même.*

De plus, il faut noter que la présence du champ `MAX_FILE_SIZE` *dans le formulaire que vous mettez à disposition n'interdit en rien un utilisateur d'utiliser son propre formulaire (sans se fixer de limitation dans la taille du fichier uploadé). Il est donc nécessaire de ne pas trop s'appuyer sur ce paramètre HTML, et de toujours vérifier dans votre programme que le fichier en question ne dépasse pas la taille autorisée.*

Le script PHP appelé par le formulaire (ici *upload.php*) dispose alors de plusieurs entrées dans la variable globale `$_FILES` de type tableau associatif (anciennement `$HTTP_POST_FILES` ou `$_POST`). Ce tableau contient une clé portant le nom spécifié dans la balise `<input type="file">` (ici `$_FILES["fichier"]`). À ce moment-là, la valeur associée est un tableau, lui aussi associatif, contenant les clés présentées dans le tableau suivant :

Tableau 9.2 : Les différentes "clés" de la variable d'environnement $_FILES dans le cas où "fichier" est le nom du champ "file" du formulaire

$_FILES	Description
`$_FILES["fichier"]["tmp_name"]`	Contient le nom temporaire (sur le serveur) du fichier qui a été transmis par le formulaire.
`$_FILES["fichier"]["name"]`	Contient le nom du fichier tel qu'il existe sur le disque du client.
`$_FILES["fichier"]["size"]`	Contient la taille en octets du fichier transmis.
`$_FILES["fichier"]["type"]`	Contient le type MIME du fichier.

register_global

Comme cela a été évoqué à de nombreuses reprises concernant d'autres variables externes, si votre fichier php.ini précise `register_global=on`*, alors* `$_FILES["fichier"]` *est directement accessible via la variable* `$fichier`*. Il est cependant fortement déconseillé de positionner* `register_global` *à* `on` *(comme précisé dans le chapitre sur les variables externes).*

Le langage PHP permet ensuite, à l'aide de quelques fonctions, de manipuler ce fichier très simplement.

La première étape peut consister en un appel à la fonction `is_uploaded_file()`. L'objectif est alors double. Il s'agit de vérifier si l'upload du fichier s'est bien déroulé (présence effective d'un fichier portant le nom indiqué), mais également de vérifier si le fichier passé en paramètre est bien un fichier qui a été téléchargé. Cette dernière vérification est très importante. Il est en effet imaginable qu'un hacker puisse tromper le script, en lui faisant croire qu'un fichier a été uploadé dans un répertoire donné sous un nom correspondant à un fichier vital de votre système. Les traitements opérés par le script s'effectueraient alors non pas sur un fichier transféré lambda, mais sur ce fichier système. Le risque est évident si vous avez opté pour une configuration avec `register_globals=on` (ce qui était la configuration par défaut avant PHP 4.2.0) et que les paramètres `GET` sont rendus globaux après les paramètres `POST` (ce qui n'est pas la configuration par défaut). Dans ce cas, il suffirait d'appeler le script de la façon suivante : *upload.php?fichier[tmp_name]=/etc/passwd* pour lui faire croire que le fichier téléchargé s'appelle */etc/passwd*. Même si le cas évoqué ici n'est pas très réaliste, dans la mesure où il résulte d'une modification aberrante du fichier de configuration, il n'est pas à exclure que d'autre moyens d'obtenir ce résultat sont envisageables. Prudence étant mère de sûreté, vous savez ce qu'il vous reste à faire…

is_uploaded_file()

Indique si le fichier est un fichier téléchargé via un formulaire HTTP de type `POST`.

Syntaxe	`boolean is_uploaded_file(string $nomFichier)`
`$nomFichier`	Chemin complet vers le fichier.
`retour`	`TRUE` si le fichier est un fichier téléchargé par formulaire, `FALSE` dans le cas contraire.

Pour mettre en évidence l'utilisation de la fonction `is_uploaded_file()` et le contenu du tableau `$_FILE`, nous utiliserons un script contenant un formulaire et qui s'appellera lui-même afin de donner les informations sur le fichier uploadé.

Listing 9.19 : is_uploaded_file.php

```
<?php
//  Vérifions que le tableau $_POST existe avant de l'utiliser
if ($_POST){
    //  Vérifions que le formulaire a été validé
    if ($_POST["envoyer"])
    {
        if (is_uploaded_file($_FILES["fichier"]["tmp_name"]))
        {
            echo "Le fichier ".$_FILES["fichier"]["name"].
                 " a bien été téléchargé.<br />";
            echo "Il fait ".$_FILES["fichier"]["size"]." et est de type ".
                  $_FILES["fichier"]["type"]."<br />";
        } else {
            echo "Le fichier ".$_FILES["fichier"]["name"].
                 " n'a pas été téléchargé<br />".
```

```
                "Peut-être dépassait-il la limite des 1Ko<br />".
                "A moins que vous n'ayez essayé de truander";
        }
    }
}
?>
<html>
    <head>
        <title>is_uploaded_file</title>
    </head>
    <body>
        <form enctype="multipart/form-data" method="post">
            <input type="hidden" name="MAX_FILE_SIZE" value="1024">
            Télécharger le fichier :
            <input name="fichier" type="file">
            <input name="envoyer" type="submit" value="Envoyer">
        </form>
    </body>
</html>
```

Une fois cette vérification effectuée, il ne reste plus qu'à déplacer ce fichier vers le dossier de travail, afin de ne pas le laisser sur le répertoire temporaire où il serait détruit à plus ou moins long terme.

Pour déplacer un fichier, il existe deux méthodes. Soit vous déplacez le fichier de façon classique à l'aide de l'instruction `move()`, soit vous déplacez le fichier à l'aide de la fonction `move_uploaded_file()`. Dans ce dernier cas, vous n'avez pas besoin d'utiliser `is_uploaded_file()`. En effet, `move_uploaded_file()` se charge de vérifier si la source est bien un fichier téléchargé à l'aide d'un formulaire.

move_uploaded_file()

Déplace un fichier téléchargé depuis le répertoire temporaire vers un dossier de destination.

Syntaxe	`boolean move_uploaded_file(string $nomFichier, string $destination)`
`$nomFichier`	Nom temporaire du fichier à déplacer.
`$destination`	Fichier de destination.
`retour`	`TRUE` si le fichier est bien un fichier uploadé et a bien été déplacé, `FALSE` s'il y a une erreur quelconque.

Il est alors possible d'écrire le script suivant, chargé de copier le fichier uploadé dans le répertoire d'exécution du script, et de lister le contenu de ce même répertoire. Ce script pourra être appelé par le formulaire *upload_form.html* présenté au début de ce chapitre.

Listing 9.20 : upload.php

```
<?php
//  Déplacement du fichier du répertoire temporaire
//  vers le répertoire courant d'où est exécuté le script.
if (move_uploaded_file($_FILES["fichier"]["tmp_name"],
                       "./".$ _FILES["fichier"]["name"]))
{
    //  Le fichier est déplacé
    //  Affichage de la liste des fichiers du répertoire
    $repertoire = opendir("./");
    while ($fichier = readdir($repertoire)) {
        echo $fichier."<br />";
    }
    closedir($repertoire);
}
?>
```

La difficulté que nous avons occultée ici est celle qui consiste à donner un nom unique au fichier. En effet, stocker le fichier sous le nom qu'il porte sur le poste client (comme cela est le cas dans l'exemple précédent) n'est généralement pas satisfaisant. Si deux utilisateurs "uploadent" un fichier *emma.jpg* et que ce fichier est copié dans le même répertoire, alors le fichier du premier sera écrasé par celui du second. Mais la stratégie à mettre en place dépend fortement du résultat que vous souhaitez obtenir. Il est envisageable de stocker le fichier sous le nom initial s'il est copié dans un répertoire propre à chaque utilisateur. Il est également possible de copier le fichier sous un nom incluant l'identifiant (unique) de l'utilisateur (si ce dernier doit s'identifier pour accéder au site).

Mais il est aussi possible d'uploader plusieurs fichiers à partir d'un unique formulaire. Il suffira, dans ce cas, de donner un nom distinct pour chaque champ de type `file`, ou bien de leur donner un nom de tableau (i.e. un nom terminé par []).

Listing 9.21 : uploadmulti_form.html

```
<html>
    <head>
        <title>Upload de plusieurs fichiers</title>
    </head>
    <body>
        <form action="uploadmulti.php" enctype="multipart/form-data"
              method="post">
            Uploader les fichiers suivants : <br />
            Fichier 0 : <input name="fichier[]" type="file" /><br />
            Fichier 1 : <input name="fichier[]" type="file" /><br />
            Fichier 2 : <input name="fichier[]" type="file" /><br />
            <input name="envoyer" type="submit" value="Envoyer">
        </form>
    </body>
</html>
```

Dans ce cas, chaque élément du tableau associatif variable `$_FILE["fichier"]` est un tableau indexé. Il ne vous reste plus qu'à récupérer les fichiers en utilisant la méthode suivante :

Listing 9.22 : uploadmulti.php

```
<?php
    //  Déplacement des fichiers du répertoire temporaire
    //  vers le répertoire courant d'où est exécuté le script.
    for ($i=0; $i<count($_FILES["fichier"]["tmp_name"]); $i++)
    {
        move_uploaded_file($_FILES["fichier"]["tmp_name"][$i],
                           "./".$_FILES["fichier"]["name"][$i]);
    }

    // Affichage de la liste des fichiers du répertoire
    $repertoire = openDir(".");
    while ($fichier = readDir($repertoire)) {
       echo $fichier."<br />";
    }
    closeDir($repertoire);
?>
```

Encore plus de fonctions d'accès au système de fichiers du serveur

Modification de l'environnement

Bien souvent, lorsque vous développez une application, le problème est de connaître le chemin du répertoire où s'exécute le programme. La commande `getcwd()` permet de récupérer le chemin du répertoire de travail.

getcwd()

Retourne le chemin du répertoire de travail.

Syntaxe	`string getcwd(void)`
`retour`	Chemin du répertoire de travail où est exécuté le script.

```
<?php
echo "Le dossier de travail est : ".getcwd();
echo "<br />";
echo "Le script est : ".$_SERVER["SCRIPT_FILENAME"];
?>
```

Et, puisque l'on peut récupérer le répertoire de travail, il peut être tout aussi intéressant de préciser ce répertoire au cours de l'exécution du programme. La commande `chdir()` permet de spécifier le dossier de travail pendant la durée de l'exécution du script.

chdir()

Modification du dossier de travail.

Syntaxe	`boolean chdir(string $repertoire)`
`$repertoire`	Nouveau répertoire de travail.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

```
<?php
if (chdir("/home/laurent/emma"))
{
    echo "Changement de répertoire effectué avec succès.";
}else{
    echo "Problème lors du changement de répertoire.<br />";
    echo "Veillez vérifier les droits d'accès au dossier.";
}
?>
```

Même si l'utilisation de ces fonctions n'est pas nécessaire pour parvenir à ce résultat, nous pouvons les utiliser pour commencer à constituer notre explorateur de fichiers, qui s'appuiera sur la fonction `explorer()` développée lors de la présentation de la fonction `readDir()`.

Listing 9.23 : gestionnaire_fichiers_01_inc.php (sans la fonction explorer)

```
<?php
function listRepertoire()
{
    // Récupération du chemin courant
    $repCourant = getcwd();

    $fichiers = explorer(".");
    // Nous ajouterons ".." (qui a été filtré par la fonction)
    if ($repCourant != "/") $fichiers = merge(array(".."), $fichiers);
?>
<table border="1" width="100%">
    <tr>
        <td><font color="#cc0000">
        <?php echo $repCourant; ?>
        </font>
        </td>
    </tr>
</table>
<table border="0" width="100%">
    for ($i=0; $i<count($fichiers); $i++)
```

```
    {
     <tr>
         <td>
<?php
        // Le fichier est-il un répertoire ?
        if (is_dir($fichiers[$i]))
        {
            /*
            Le fichier est un répertoire
            On affiche alors un lien qui va nous permettre
            de visualiser le contenu de ce dossier
            */
?>
            <a href="?repertoire=<?php echo $repCourant."/".$fichiers[$i];?>">
            <?php echo $fichiers[$i];?>
            </a>
<?php
        }else{
            /*
            Le fichier n'est pas un répertoire
            On affiche simplement le nom du fichier
            */
            echo $fichiers[$i];
        }
?>
        </td>
    </tr>
<?php
    }
?>
    echo "</table>";
}
```

Le programme principal sera alors (pour l'instant) :

Listing 9.24 : gestionnaire_fichier_01.php

```
<?php
include("gestionnaire_fichier_01_inc.php");

//  Vérifie si "repertoire" est passé en paramètre.
if ($_GET["repertoire"])
{
    if (!@chdir($_GET["repertoire"]))
    {
        echo "Le changement de repertoire a échoué.";
    }
}
listeRepertoire(".");
```

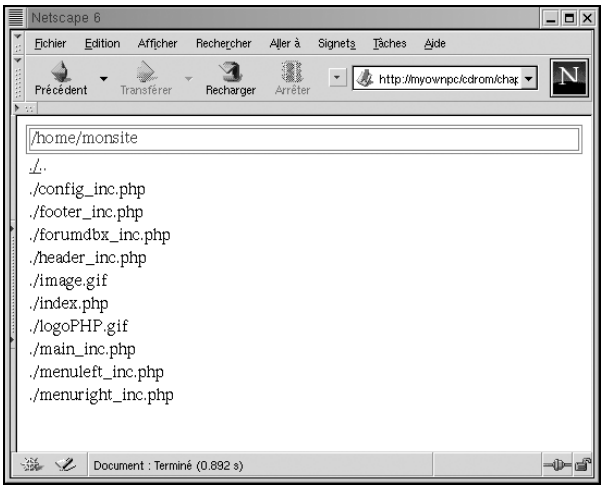


Figure 9.7 : *Gestionnaire de fichiers (version 1)*

Sécurité du système
Maintenant que l'utilisateur peut se déplacer simplement dans la liste des répertoires, il est important de rendre inaccessibles certains dossiers contenant des informations que vous ne voulez pas rendre publiques. Modifier les droits d'accès du serveur web sur les répertoires en question permet d'interdire le changement de dossier de travail.
Exemple : `rwxrw----  utilisateur  apache`

Statistiques sur les fichiers

Chemin d'accès aux fichiers

Avant d'ouvrir un fichier, il est utile, voire nécessaire, de connaître certaines informations concernant son chemin d'accès, son nom, etc. Le langage PHP possède quelques instructions pouvant être très pratiques pour récupérer ces informations.

Il est simple de récupérer le nom d'un fichier (sans le chemin) à l'aide de l'instruction `basename()`.

basename()

Extrait d'une chaîne de caractères la partie correspondant au nom du fichier.

Syntaxe	`string basename(string $cheminFichier [, string $suffixe])`
`$cheminFichier`	Chemin d'un fichier (qui n'a pas besoin d'exister).
`$suffixe`	Ce paramètre optionnel indique s'il faut supprimer le suffixe.
`retour`	Retourne le nom du fichier.

```
<?php
$image= "/home/laurent/emma.png";
//  Affichage de emma.png
echo basename($image);
echo "<br />";
//  Affichage de emma
echo basename($image, "png");
?>
emma.png
emma
```

Pour aller dans le même sens que `basename()`, on va maintenant étudier l'instruction `dirname()` qui retourne le chemin du répertoire contenant un fichier.

dirname()

Extrait d'une chaîne de caractères la partie correspondant au chemin.

Syntaxe	`string dirname(string $cheminFichier)`
`$cheminFichier`	Chemin d'un fichier (qui n'a pas besoin d'exister).
`retour`	Retourne le chemin du dossier contenant le fichier.

```
<?php
$image= "/home/laurent/emma.png";
//  Affichage de /home/laurent
echo basename($image);
?>
/home/laurent
```

Une fonction permet également de récupérer toutes ces informations en une seule fois. L'instruction `pathInfo()` retourne un tableau contenant les différents renseignements sur un chemin de fichier.

pathInfo()

Extrait d'une chaîne de caractères les parties correspondant au chemin, au nom du fichier (extension incluse) et à l'extension.

Syntaxe	`array pathInfo(string $cheminFichier)`
`$cheminFichier`	Chemin d'un fichier (qui n'a pas besoin d'exister).
`retour`	Tableau associatif contenant les clés : "`dirname`" associée au nom du répertoire. "`basename`" associée au nom du fichier. "`extension`" associée à l'extension du fichier.

```
<?php
$infoChemin = pathInfo("/home/laurent/emma.png");
while (list ($key, $val) = each ($infoChemin)) {
    echo "$key = $val<br />";
}
?>
dirname = /home/laurent
basename = emma.png
extension = png
```

Grâce à la fonction `realPath()`, il est possible d'obtenir le chemin absolu d'un fichier à partir de son chemin relatif.

realPath()

Retourne le chemin absolu d'un fichier.

Syntaxe	`string realPath(string $cheminFichier)`
`$cheminFichier`	Chemin d'un fichier.
`retour`	Chemin absolu du fichier.

```
<?php
echo realpath("../../info.php");
echo "<br />";
echo realpath("bonzour.jpg");
?>
/home/e-smith/files/ibays/kangouroo/html/info.php
/home/e-smith/files/ibays/kangouroo/html/bible/fichier/bonzour.jpg
```

Nature du fichier

Une série de fonctions vous permet de vérifier la nature du fichier ou du répertoire, à savoir : `is_dir()`, `is_executable()`, `is_file()`, `is_link()`, `is_readable()`, `is_writable()` (ou son alias `is_writeable`) et `is_uploaded_file()`.

is_dir()

Indique si le fichier passé en paramètre est un répertoire ou non.

Syntaxe	`boolean is_dir(string $nomFichier)`
`$nomFichier`	Chemin du fichier.
`retour`	`TRUE` si le fichier est un répertoire, `FALSE` dans le cas contraire.

```
<?php
if (is_dir("indetermine"))
{
    echo "OUUIII, je suis un repertoire !";
}else{
    echo "Ben non je suis autre chose, snif !";
}
?>
```

is_file()

Indique si le fichier est un fichier classique.

Syntaxe	`boolean is_file(string $nomFichier)`
`$nomFichier`	Chemin du fichier.
`retour`	`TRUE` si le fichier est un fichier classique, `FALSE` dans le cas contraire.

```
<?php
if (is_file("indetermine"))
{
    echo "Je suis bien un fichier !";
}else{
    echo "C'est pas encore ça, je ne suis pas un fichier.";
}
?>
```

is_link() (inopérant sous Windows)

Indique si le fichier est un lien symbolique ou non.

Syntaxe	`boolean is_link(string $nomFichier)`
`$nomFichier`	Chemin du fichier.
`retour`	`TRUE` si le fichier est un lien symbolique, `FALSE` dans le cas.

```
<?php
if (is_link("indetermine"))
{
    echo "C'est ça ! Je suis un lien symbolique.";
}else{
    echo "Ben, toujours pas :o(.";
}
?>
```

Vous pouvez aussi remplacer ces précédentes fonctions par une autre bien pratique : la fonction `fileType()` renvoie une chaîne de caractères indiquant le type du fichier.

fileType()

Retourne une chaîne indiquant le type du fichier passé en paramètre.

Syntaxe	`string filetype(string $nomFichier)`
`$nomFichier`	Chemin du fichier.
`retour`	Retourne le type du fichier à savoir : "`fifo`". "`char`". "`dir`" s'il s'agit d'un répertoire. "block". "`link`" s'il s'agit d'un lien symbolique. "`file`" s'il s'agit d'un véritable fichier. "`unknown`" dans le cas ou le type de "fichier" est inconnu.

```
<?php
echo filetype("monfichier.txt");
?>
```

Droits sur les fichiers

Différentes fonctions permettent également de tester les droits sur les fichiers. Dans les cas suivants, les droits sont testés par rapport à l'utilisateur et au groupe exécutant le script. Ce sera la plupart du temps l'utilisateur du serveur HTTP (généralement `apache` ou `nobody`, par exemple, dans le cas d'un serveur Apache). Ces fonctions sont très intéressantes si vous voulez développer une installation automatique d'une de vos applications, à la manière de SPIP par exemple (voir les annexes pour l'installation de SPIP). Votre installation devant créer ou retoucher un fichier de configuration, il peut être utile de vérifier si celui-ci est accessible en écriture par votre script. Si c'est le cas, vous exécutez l'installation, sinon vous envoyez des informations à l'utilisateur indiquant pas à pas ce qu'il doit faire pour que le script puisse être lancé.

is_readable()

Indique si le fichier est accessible en lecture.

Syntaxe	`boolean is_readable(string $nomFichier)`
`$nomFichier`	Chemin du fichier.
`retour`	`TRUE` si le fichier est accessible en lecture, `FALSE` dans le cas contraire.

```
<?php
if (is_readable("monfichier.txt"))
{
    echo "Je suis un livre ouvert.";
}else{
    echo "Vous ne regarderez pas en moi.";
}
?>
```

is_writable()

Indique si le fichier est accessible en écriture.

Syntaxe	`boolean is_writable(string $nomFichier)`
`$nomFichier`	Chemin du fichier.
`retour`	`TRUE` si le fichier est accessible en écriture, `FALSE` dans le cas contraire.

```
<?php
if (is_writable("monfichier.txt"))
{
    echo "Vous pouvez me modifier sans problème.";
}else{
    echo "Ne me touchez pas !";
}
?>
```

Vous pouvez également utiliser la fonction `is_writeable()` qui est un alias de `is_writable()`.

is_executable()

Indique si le fichier est exécutable ou non.

Syntaxe	`boolean is_executable(string $nomFichier)`
`$nomFichier`	Chemin du fichier.
`retour`	`TRUE` si le fichier est exécutable, `FALSE` dans le cas contraire.

```
<?php
if (is_executable("monfichier.txt"))
{
    echo "Je marche avec vous.";
}else{
    echo "Vous pouvez toujours rêver pour m'exécuter !";
}
?>
```

Existence et taille des fichiers

Le langage PHP possède une instruction simple pour vérifier l'existence ou non d'un fichier. Il est souhaitable de tester si ce fichier est bien à l'endroit où il devrait être avant de lui appliquer une ou plusieurs fonctions de statistiques ; c'est ce que permet la fonction `file_exists()`.

file_exists()

Permet de vérifier qu'un fichier existe bien.

Syntaxe	`boolean file_exists(string $nomFichier)`
`$nomFichier`	Chemin du fichier à tester.
`retour`	`TRUE` si le fichier est bien présent, `FALSE` s'il n'existe pas ou si une erreur s'est produite.

```
<?php
if (file_exists("emma.png")==TRUE)
{
    echo "Le fichier emma.png est bien présent sur le disque.";
} else {
    echo "Le fichier emma.png n'existe pas.";
}
?>
```

Pour continuer notre chemin et compléter notre petit gestionnaire de fichiers en ligne, nous devons également récupérer la taille du fichier. La fonction qui va nous permettre de réaliser cela est `fileSize()`.

fileSize()

Récupère la taille du fichier passé en paramètre en octets.

Syntaxe	`int fileSize(string $nomFichier)`
`$nomFichier`	Chemin du fichier.
`retour`	Retourne la taille du fichier en octets.

Ceci nous permet de créer une nouvelle fonction chargée de récupérer la taille du fichier et de l'afficher dans un format adapté (i.e. Ko, Mo, etc. selon les cas).

Listing 9.25 : gestionnaire_fichier_02_inc.php (extrait)

```
<?php
function tailleFichier($fichier)
{
    //  calculs des taux de conversion entre Ko,
    //  Mo et octet
    $Ko = pow(2, 10);
    $Mo = pow(2, 20);

    //  recupère la taille du fichier en octets
    $taille = fileSize($fichier);
```

```
    //  Pas de conversion
    if ($taille<$Ko){
        $tailleDef = $taille;
    //  Conversion en Ko
    } elseif ($taille>=$Ko && $taille<$Mo){
        $tailleDef = round($taille/$Ko, 1)."k";
    //  Conversion en Mo
    } else {
        $tailleDef = round($taille/$Mo, 1)."M";
    }
    return $tailleDef;
}
?>
```

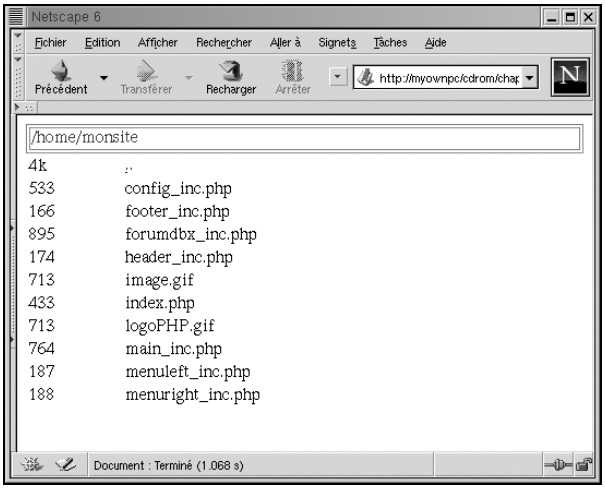


Figure 9.8 : *Gestionnaire de fichiers (version 2)*

Utilisateurs et groupes propriétaires des fichiers

Afin de récupérer l'identifiant de l'utilisateur et le groupe propriétaire du fichier, nous utiliserons deux autres fonctions qui sont `fileOwner()` et `fileGroup()`. Ces fonctions renvoient les UID et GID correspondant au fichier.

fileOwner() (inopérant sous Windows)

Retourne l'identifiant de l'utilisateur (UID) propriétaire du fichier.

Syntaxe	`int fileOwner(string $nomFichier)`
`$nomFichier`	Chemin du fichier.
`retour`	Identifiant de l'utilisateur propriétaire du fichier (`0` sous Windows), `FALSE` en cas d'erreur.

fileGroup() (inopérant sous Windows)

Retourne l'identifiant du groupe (GID) propriétaire du fichier.

Syntaxe	`int fileGroup(string $nomFichier)`
`$nomFichier`	Chemin du fichier.
`retour`	Identifiant du groupe propriétaire du fichier (`0` sous Windows), `FALSE` en cas d'erreur.

```
<?php
function proprietaire($fichier)
{
    $utilisateur = fileowner($fichier);
    $groupe      = filegroup($fichier);
    return array("utilisateur"=>$utilisateur,
                 "groupe"=>$groupe);
}
$tabPropietaire = proprietaire("fichier.txt");
echo "utilisateur = ".$tabPropietaire["utilisateur"];
echo "<br />";
echo "groupe = ".$tabPropietaire["groupe"];
?>
utilisateur = 101
groupe = 102
```

Il peut être intéressant d'afficher l'utilisateur et le groupe par leurs noms respectifs. Il faut, dans ce cas, utiliser les fonctions du module `POSIX`, à savoir `posix_getpwuid()` et `posix_getgrgid()`. La première fonction renvoie les informations sur l'utilisateur possédant l'UID passé en paramètre, et la seconde retourne les informations sur le groupe possédant le GID donné.

Vous pouvez vous reporter au chapitre "Les processus et les identifiants" pour plus d'informations sur `posix_getpwuid()` et `posix_getgrgid()`.

Le script suivant retourne donc les noms du propriétaire et du groupe du fichier, et sera intégré à notre gestionnaire de fichiers.

Listing 9.26 : gestionnaire_fichier_03_inc.php (extrait)

```
<?php
function proprietaire($fichier)
{
    //  recupère l'UID du fichier
    $uid = fileowner($fichier);
    //  Recupère les informations sur l'UID
    $tabUtilisateur = posix_getpwuid($uid);
```

```
    //  recupère le GID du fichier
    $gid = filegroup($fichier);
    //  Récupère les informations sur le GID
    $tabGroup = posix_getgrgid($gid);

    //  Renvoie les noms du groupe et de l'utilisateur
    return array(
        "utilisateur"=>$tabUtilisateur["name"],
        "groupe"=>$tabGroup["name"]
        );
}
?>
```

Simplement appliqué à un fichier de la façon suivante :

```
<?php
$tabPropietaire = proprietaire("fichier.txt");
echo "utilisateur = ".$tabPropietaire["utilisateur"];
echo "<br />";
echo "groupe = ".$tabPropietaire["groupe"];
?>
```

cela pourrait donner :

utilisateur = admin
groupe = www

Utilisé dans notre gestionnaire de fichiers, cela donne :

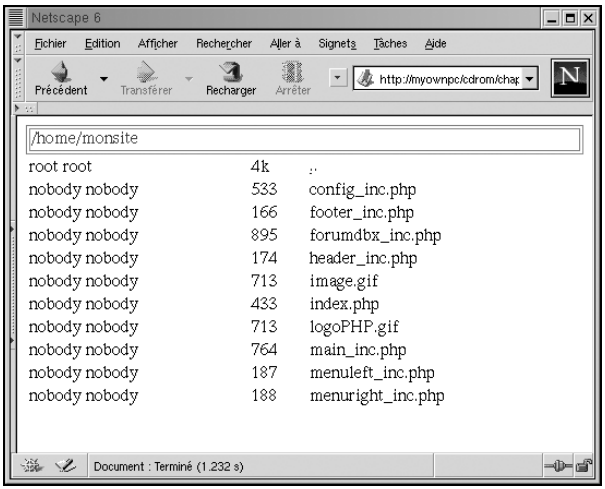


Figure 9.9 : *Gestionnaire de fichiers (version 3)*

Dates et heures fichiers

Il peut être intéressant de récupérer la date de la dernière modification du fichier. On utilisera alors la fonction `filemtime()` du langage PHP.

filemtime()

Retourne la date de la dernière modification du fichier.

Syntaxe	`int filemtime(string $nomFichier)`
`$nomFichier`	Chemin du fichier.
`retour`	Date de la dernière modification en secondes depuis epoch (1er janvier 1970).

Voilà une nouvelle fonction que nous pouvons ajouter à notre gestionnaire de fichiers en l'agrémentant d'une mise en forme de la date au format mois/jour/heure/minutes.

Listing 9.27 : gestionnaire_fichier_04_inc.php (extrait)

```
<?php
function modificationFichier($fichier)
{
    //  Récupération de la dernière modification du fichier
    $modification = filemtime($fichier);

    //  Retourne la date de modifications formatée
    return strftime("%b %d %H:%M", $modification)."\n";
}
?>
```

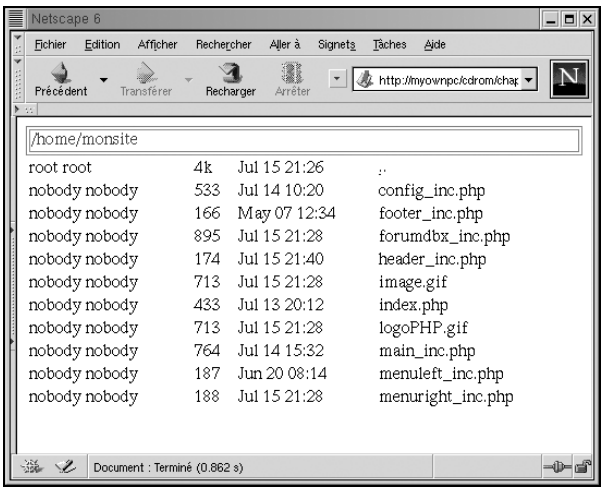


Figure 9.10 : *Gestionnaire de fichiers (extrait 4)*

Pour modifier cette date, vous pouvez utiliser la fonction `touch()`.

touch()

Spécifie une nouvelle date de modification pour un fichier (crée le fichier s'il n'existe pas).

Syntaxe	`boolean touch(string $nomFichier [, int $date])`
`$nomFichier`	Chemin du fichier dont vous souhaitez modifier la date de dernière modification.
`$date`	Nouvelle date de modification (en secondes depuis epoch). Par défaut, il s'agit de la date courante.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

```
<?php
//  Retourne la date de dernière modification du fichier
echo filemtime("fichier.txt");
//  Change la date dans la date courante
touch("fichier.txt");
//  Retourne la nouvelle date de modification
echo filemtime("fichier.txt");
?>
```

De la même façon, il est possible d'accéder à la date de dernier accès au fichier, ainsi qu'à la date du dernier accès à l'inode (dernière fois que les droits, utilisateur, groupe, date de dernière modification, etc. du fichier ont été modifiés). Les fonctions `fileatime()` et `filectime()` permettent au développeur d'accéder à ces différentes informations.

fileatime()

Retourne la date et l'heure du dernier accès au fichier.

Syntaxe	`int fileatime(string $nomFichier)`
`$nomFichier`	Chemin du fichier.
`retour`	Date du dernier accès au fichier (en secondes depuis epoch).

```
<?php
function dateAcces($fichier)
{
    //  Récupération de la dernière modification du fichier
    $modification = fileatime($fichier);

    //  Retourne la date de modifications formatée
    return strftime("%b %d %H:%M", $modification)."\n";
}
echo "Dernier accès au fichier le ". dateAcces("fichier.txt");
?>
```

filectime()

Retourne la date et l'heure du dernier accès à l'inode.

Syntaxe	`int filectime(string $nomFichier)`
`$nomFichier`	Chemin du fichier.
`retour`	Retourne la date du dernier accès à l'inode (en secondes depuis epoch).

```
<?php
function dateAccesInode($fichier)
{
    //  Récupération de la dernière modification du fichier
    $modification = filectime($fichier);

    //  Retourne la date de modifications formatée
    return strftime("%b %d %H:%M", $modification)."\n";
}
echo "Dernier accès à l'inode le ". dateAccesInode("fichier.txt");
?>
```

Date de modification et d'accès et liens symboliques
Si vous utilisez l'une des fonctions `filectime()`, `fileatime()` ou `filemtime()` sur un lien symbolique, vous aurez la date de modification du fichier sur laquelle pointe le lien. Pour récupérer les informations sur le lien symbolique, vous devez utiliser la fonction `lstat()`.

Inode

Comme nous l'avons évoqué précédemment, l'inode (*index node* ou nœud d'index) est une structure contenant les différentes informations liées à un fichier (droits, propriétaire, date de dernière modification, etc). À chaque inode est associé un identifiant INumber qui est le numéro d'index de l'inode.

Le langage PHP possède une fonction qui permet de récupérer ce numéro d'index pour un fichier donné. L'instruction `fileinode()` permet en effet de retourner l'INumber.

fileInode() (inopérant sous Windows)

Retourne l'INumber de l'inode d'un fichier.

Syntaxe	`int fileInode(string $nomFichier)`
`$nomFichier`	Chemin du fichier.
`retour`	Inode du fichier (`0` sous Windows).

```
<?php
echo fileinode("emma.png");
?>
```

La fonction qui fait tout

Pour récupérer les différentes informations en une seule fois, le langage PHP possède une fonction très utile. La fonction `stat()` permet, en effet, de retrouver tous les paramètres du fichier dans un tableau. Cette instruction utilise la structure des fichiers pour obtenir les informations nécessaires.

stat()

Retourne les informations à propos d'un fichier dans un tableau.

Syntaxe	`array stat(string $nomFichier)`
`$nomFichier`	Chemin du fichier à analyser.
`retour`	Tableau à la fois indexé et associatif contenant toutes les informations du fichier.

Le tableau retourné contient toutes les informations relatives à un fichier ; ce tableau peut se décomposer comme suit :

Tableau 9.3 : Le tableau retourné par la fonction stat()

Indice	Clé	Description
0	`Dev`	Numéro d'identifiant du système de fichiers.
1	`Ino`	INumber de l'inode représentant la structure du fichier.
2	`Mode`	Les droits d'accès sur le fichier (valeur en décimal).
3	`Nlink`	Nombre de liens vers le fichier.
4	`Uid`	Identifiant du propriétaire du fichier.
5	`Gid`	Identifiant du groupe propriétaire du fichier.
6	`Rdev`	Identifiant du matériel (disque dur, CD-ROM, etc.) contenant le fichier (inode) (`-1` sous Windows).
7	`size`	Taille du fichier en octets.
8	`atime`	Indique la date du dernier accès en secondes depuis epoch (sous UNIX, epoch correspond au 1er janvier 1970 à 0 heure 0 minute et 0 seconde, sous MacOS c'est 1er janvier 1904 à 0 heure 0 minute et 0 seconde qui sert de référence).
9	`mtime`	Date de la dernière modification du fichier depuis epoch.
10	`ctime`	Date du dernier changement (modification de l'inode).

Indice	Clé	Description
11	`blksize`	Taille des blocs (en octets) de référence pour les entrées/sorties du système de fichier (`-1` sous Windows).
12	`blocks`	Nombre de blocs alloués pour le fichier (`-1` sous Windows).

L'exemple suivant,

```
<?php
function statistique($cheminFichier)
{
    $tableau = stat($cheminFichier);
    while (list($key, $val) = each($tableau)) {
        echo "$key : $val<br>";
    }
}
statistique("fichier.txt");
?>
```

pourrait retourner :

```
0 : 834
1 : 131934
2 : 33264
3 : 1
4 : 101
5 : 102
6 : 0
7 : 23
8 : 1023545447
9 : 1023545460
10 : 1023545460
11 : 4096
12 : 8
dev : 834
ino : 131934
mode : 33264
nlink : 1
uid : 101
gid : 102
rdev : 0
size : 23
atime : 1023545447
mtime : 1023545460
ctime : 1023545460
blksize : 4096
blocks : 8
```

S'il faut simplement récupérer les droits correspondant à un fichier, il suffit alors d'écrire le programme suivant :

```
<?php
$tableau = stat("fichier.txt");
//  recupère les permissions sur le fichier
$droits = $tableau["mode"];
//  Conversion en octale
$octalPerm = decoct($droits);
//  Récupération des 3 derniers caractères
$normPerm = substr($octalPerm,-3);
echo $normPerm;
?>
```


REMARQUE

Retrouver le type du fichier

Vous remarquerez que, dans l'exemple précédent, nous avons récupéré simplement les trois derniers caractères. En effet, seuls les derniers caractères correspondent aux permissions du fichier. Les autres indiquent le type du fichier.

0x1000 Port (Named pipe).
0x2000 Correspond à un fichier de type caractère (imprimante, port série, clavier, souris,..).
0x4000 Répertoire.
0x6000 Représente un matériel disposant d'un système d'entrée/sortie (disques durs IDE, RAM, etc.).
0x8000 Fichier dit "normal".
0xA000 Lien symbolique.
0xC000 Socket.

Appliquée à un lien symbolique, la fonction `stat()` retournera les informations concernant le fichier pointé. Si vous souhaitez obtenir les informations concernant le lien, vous devrez faire appel à la fonction `lstat()`. Cette dernière est identique à la fonction `stat()` et retourne un tableau identique.

lstat()

Retourne les informations à propos d'un lien symbolique (ou d'un fichier).

Syntaxe	`array lstat(string $nomFichier)`
`$nomFichier`	Chemin du lien symbolique (ou fichier) à analyser.
`retour`	Tableau contenant toutes les informations du fichier.

Vous pouvez sans doute être amené à rechercher le fichier pointé par le lien symbolique. Il est alors nécessaire d'utiliser la fonction `readLink()`.

readLink() (non disponible sous Windows)

Retourne le nom et le chemin du fichier cible du lien symbolique.

Syntaxe	`string readLink(string $nomLienSymb)`
`$nomLienSymb`	Lien symbolique à analyser.
`retour`	Chemin du fichier pointé par le lien symbolique (`FALSE` accompagné du message de type "warning" sous Windows).

```
<?php
if (symlink("/home/laurent/emma.png",
            "/home/damien/emma.png") {
    echo "Le lien symbolique a bien été créé.";
} else {
    echo "Le lien symbolique n'a pas été créé.";
}
echo readlink("/home/damien/emma.png");
?>
/home/laurent/emma.png
```

Si vous voulez simplement tester l'existence d'un lien symbolique, vous pouvez utiliser la commande `linkInfo()`. En effet, la commande `file_exists()` retournera l'existence du fichier cible et non celle du lien. Remarquez que, si l'instruction trouve le fichier cible, c'est que le lien existe forcément. Mais il peut s'avérer que le fichier cible soit inexistant.

linkInfo() (non disponible sous Windows)

Retourne l'identifiant du matériel où est stocké le lien symbolique.

Syntaxe	`int linkInfo(string $nomLienSymb)`
`$nomLienSymb`	Lien symbolique à analyser.
`retour`	Identifiant du matériel (ou *device*, c'est-à-dire le matériel où est stocké le lien symbolique ; attention, *device* ne fait pas référence à un disque dur, mais à une partition) qui héberge le lien, ou `FALSE` si le lien n'existe pas, ou si une erreur se produit (avec un message de type "warning" sous Windows).

```
<?php
if (@linkInfo("emma.png") != -1) {
    echo "Le lien symbolique existe bien.";
} else {
    echo "Le lien symbolique n'existe pas.";
}
?>
```

Il est aussi possible d'utiliser `fstat()` afin d'obtenir la liste des informations sur un fichier. La différence avec la fonction `stat()` étant que `fstat()` ne prend pas en paramètre le chemin d'un fichier, mais un pointeur sur ce fichier.

fstat()

Retourne les informations à propos d'un fichier dans un tableau.

Syntaxe	`array fstat(string $fp)`
`$fp`	Pointeur sur le fichier à analyser tel que retourné par `fopen()`.
`retour`	Tableau à la fois indexé et associatif contenant toutes les informations du fichier (comme le propose la fonction `stat()`), ou `FALSE` si une erreur survient.

```
<?
function statistique($cheminFichier)
{
    $fp = fopen("$cheminFichier","r");
    $tableau = fstat($fp);
    while (list($key, $val) = each($tableau)) {
        echo "$key : $val<br>";
    }
    fclose($fp);
}
statistique("fichier.txt");
?>
0 : 834
1 : 131934
2 : 33264
3 : 1
4 : 101
5 : 102
6 : 0
7 : 23
8 : 1023545447
9 : 1023545460
10 : 1023545460
11 : 4096
12 : 8
dev : 834
ino : 131934
mode : 33264
nlink : 1
uid : 101
gid : 102
rdev : 0
size : 23
atime : 1023545447
```

```
mtime : 1023545460
ctime : 1023545460
blksize : 4096
blocks : 8
```

Les différentes fonctions de statistiques demandent beaucoup de ressources au système. C'est pourquoi il existe un système de cache qui permet de conserver, pendant la durée de l'exécution du programme, les différentes valeurs relatives à un fichier. En conséquence, toutes les modifications apportées au fichier après un appel aux fonctions de type `stat()` ne seront pas visibles des appels ultérieurs à ces mêmes fonctions. Afin de renouveler le cache, vous devez faire appel à la fonction `clearStatCache()`. Le cache n'est valide que pendant la durée de l'exécution du programme ; il n'est donc pas nécessaire d'appeler la fonction `clearStatCache()` avant chaque statistique sur les fichiers.

clearStatCache()

Réinitialise le cache des fonctions de statistiques.

Syntaxe `void clearStatCache(void)`

```
<?php
//  Initialisation des droits en 770
chmod("fichier.txt", 0770);

//  Lecture des droits
$tableau = stat("fichier.txt");
echo $tableau["mode"]."<br />";

//  modification des droits sur le fichier en 777
chmod("fichier.txt", 0777);

//  Lecture des droits
$tableau = stat("fichier.txt");
echo $tableau["mode"]."<br />";

//  Réinitialisation du cache
clearstatcache();

//  Lecture des droits
$tableau = stat("fichier.txt");
echo $tableau["mode"]."<br />";
?>
33272
33272
33279
```

Nous pouvons observer, dans l'exemple précédent, que la modification des droits n'entraîne pas de modification dans l'affichage des statistiques du fichier. Ce qui démontre la nécessité de l'appel à la fonction `clearStatCache()`. Le cache est vidé, et un nouvel appel à la fonction `stat()` affiche les droits corrects du fichier.

Comme nous l'avons annoncé précédemment, les fonctions de statistiques entraînent une charge plus importante pour le système. Ceci est d'autant plus vrai lorsque vous utilisez les fonctions `stat()`, `fstat()` et `lstat()` qui retournent beaucoup d'informations. Certaines fonctions, plus légères, peuvent les remplacer facilement si vous désirez seulement récupérer certaines informations sur le fichier.

Pour récupérer les permissions sur un fichier, il est plus judicieux d'utiliser la fonction `filePerms()` du langage PHP.

filePerms()

Retourne les permissions du fichier.

Syntaxe	`int filePerms(string $nomFichier)`
`$nomFichier`	Chemin du fichier.
`retour`	Droits d'accès sur le fichier (sous forme décimale) ou `FALSE` en cas d'erreur.

```
<?php
$droits = fileperms("fichier.txt");
//  Conversion en octale
$octalPerm = decoct($droits);
//  Récupération des 3 derniers caractères
$normPerm = substr($octalPerm,-3);
echo $normPerm;
?>
```

Nous pouvons améliorer notre gestionnaire de programmes pour qu'il affiche les droits des fichiers sous une forme classique pour un utilisateur Unixien.

Listing 9.28 : gestionnaire_fichiers_05_inc.php (extrait)

```
<?php
function uPerm($perm)
{
    switch ($perm)
    {
        case 0:
            $retPerm = "---";
            break;
        case 1:
            $retPerm = "--x";
            break;
        case 2:
            $retPerm = "-w-";
            break;
        case 3:
            $retPerm = "-wx";
```

```
            break;
        case 4:
            $retPerm = "r--";
            break;
        case 5:
            $retPerm = "r-x";
            break;
        case 6:
            $retPerm = "rw-";
            break;
        case 7:
            $retPerm = "rwx";
            break;
    }
    return $retPerm;
}

function affichePermission($fichier)
{
    //  Récupère le mode du fichier et conversion en octal
    $mode = fileperms($fichier);

    //  Détermine le type du fichier
    if(($mode & 0x1000) === 0x1000)
        $type = "p";    //  Port
    elseif(($mode & 0x2000) === 0x2000)
        $type = "c";    //  Matériel
    elseif(($mode & 0x4000) === 0x4000)
        $type = "d";    //  Répertoire
    elseif(($mode & 0x6000) === 0x6000)
        $type = "b";    //  Matériel FIFO
    elseif(($mode & 0x8000) === 0x8000)
        $type = "-";    //  Fichier normal
    elseif(($mode & 0xa000) === 0xa000)
        $type = "l";    //  Lien symbolique
    elseif(($mode & 0xc000) === 0xc000)
        $type = "s";    //  Socket
    else
        $type = "u";    // Unknown

    $mode = decoct($mode);
    $perTemp = substr($mode,-3);

    $permission["utilisateur"] = uPerm(substr($perTemp,0,1));
    $permission["groupe"] = uPerm(substr($perTemp,1,1));
    $permission["tous"] = uPerm(substr($perTemp,2,1));

    return $type.$permission["utilisateur"].
        $permission["groupe"].$permission["tous"];

}
?>
```

L'exécution de ce programme affiche donc une réponse de la forme `drwxrwxrwx`.

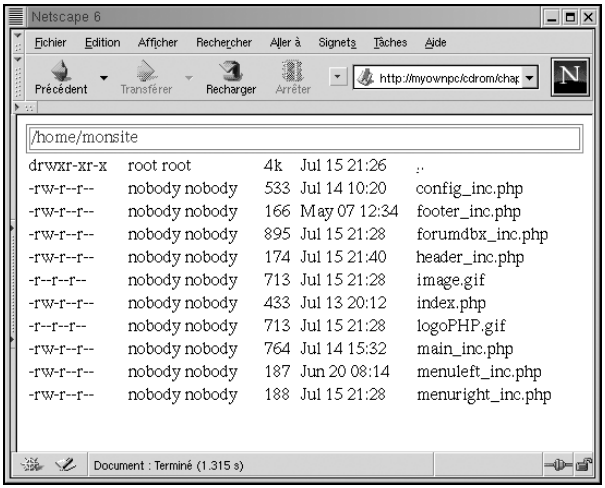


Figure 9.11 : *Gestionnaire de programmes (version 5)*

Informations sur le disque

Nous allons à présent compléter ce petit gestionnaire de fichiers en affichant l'espace disque disponible : l'espace disque total. `disk_free_space()`, ou son alias `diskfreespace()`, retourne la place restante sur le serveur ; l'instruction `disk_total_space()` renvoie la capacité totale du disque.

disk_free_space()

Retourne l'espace disponible sur le disque contenant un répertoire donné.

Syntaxe	`float disk_free_space(string $repertoire)`
`$repertoire`	Chemin du dossier où l'espace est à analyser.
`retour`	Espace disponible en octets, ou `FALSE` en cas d'erreur.

```
<?php
//  Espace disponible dans le répertoire courant.
echo disk_free_space("./");
echo "<br />";
//  Espace disponible à la racine.
echo disk_free_space("/");
?>
```

disk_total_space()

Retourne l'espace total du disque contenant un répertoire donné.

Syntaxe	`float disk_total_space(string $repertoire)`
`$repertoire`	Chemin du dossier où l'espace est à analyser.
`retour`	Espace total en octets, ou `FALSE` en cas d'erreur.

```
<?php
//  Espace disque total dans le repertoire courant.
echo disk_total_space("./");
echo "<br />";
//  Espace disque total à la racine.
echo disk_total_space("/");
?>
```

Il est à présent possible d'ajouter une fonction utilisateur à notre gestionnaire de fichiers. Celle-ci va nous permettre d'afficher l'espace restant ainsi que l'espace total dans le répertoire courant.

Listing 9.29 : gestionnaire_fichier_06.php (extrait)

```
function espaceDisque($repCourant)
{
    $Mo = pow(2, 20);
    //  Retourne une chaîne de caractères indiquant
    //  l'espace disque disponible sur l'espace disque
    //  total.
    $chDisque = round(disk_free_space($repCourant)/$Mo, 1)."Mo";
    $chDisque .= " / ";
    $chDisque .= round(disk_total_space($repCourant)/$Mo, 1)."Mo";

    return $chDisque;
}
```

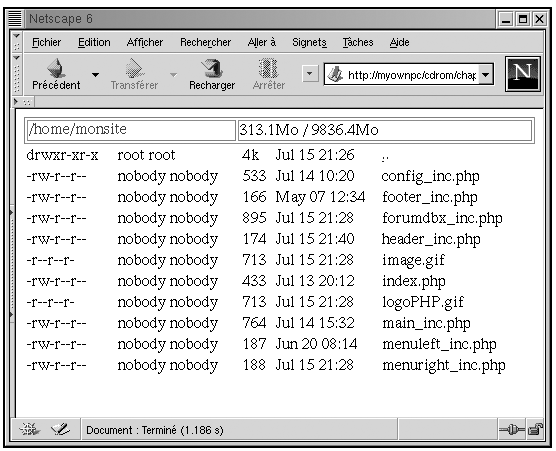


Figure 9.12 : *Gestionnaire de fichiers (version 6)*

Exemple d'application

À présent, nous pouvons modifier notre gestionnaire de fichiers de façon à permettre aux utilisateurs d'accéder à la copie de fichiers, ainsi qu'aux autres fonctions : création de liens, de liens symboliques, de répertoires, suppression et déplacement de fichiers.

Nous allons maintenant ajouter des boutons radio afin de permettre la sélection d'un des fichiers et d'une série de champs texte, ainsi que des boutons destinés à sélectionner une opération à effectuer en précisant éventuellement un nom.

Ce qui nous donne le script complet final suivant :

Listing 9.30 : gestionnaire_fichiers_07_inc.php

```
<?php
// La fonction d'exploration
// $chemin : Répertoire à explorer
// $recursif : TRUE si l'exploration doit être récursive
// $filtre : Expression régulière de filtrage des fichiers

function explorer($chemin, $recursif=FALSE, $filtre=NULL) {
    $listeFichier = array();

    $repertoire = openDir($chemin);
    while ($fichier = readDir($repertoire)) {
        // Inutile de tenir compte des entrées . et ..
        if (($fichier != ".")&&($fichier != "..")) {
            // Est-ce que $file est un répertoire ?
            // Pour le savoir il suffit d'appeler is_dir
            // mais attention n'oublions pas d'ajouter
            // le chemin au nom du fichier
            if (is_dir($chemin."/".$fichier)&&($recursif)) {
                // oui ? alors explorons-le
                // et ajoutons le résultat à la liste de fichiers
                $listeFichier = array_merge($listeFichier,
                                            explorer($chemin."/".$fichier,
                                            $recursif, $filtre));
            } else {
                // sinon, c'est un fichier et s'il répond
                // aux critères de l'expression régulière
                // on l'ajoute a la liste des fichiers
                if (is_null($filtre)||eregi($filtre, $fichier)) {
                    $listeFichier[] = $chemin."/".$fichier;
                }
            }
        }
    }
    // C'est fini. On ferme !
    closeDir($repertoire);
    // et on retourne le résultat trié
    sort($listeFichier);
    return $listeFichier;
}
```

```
//  Retourne la taille du fichier
function tailleFichier($fichier)
{
    //  calculs des taux de conversion entre Ko, Mo et octet
    $Ko = pow(2, 10);
    $Mo = pow(2, 20);

    //  recupère la taille du fichier en octets
    $taille = fileSize($fichier);

    //  Pas de conversion
    if ($taille<$Ko){
        $tailleDef = $taille;
    //  Conversion en Ko
    } elseif ($taille>=$Ko && $taille<$Mo){
        $tailleDef = round($taille/$Ko, 1)."k";
    //  Conversion en Mo
    } else {
        $tailleDef = round($taille/$Mo, 1)."M";
    }
    return $tailleDef;
}

//  Retourne le groupe et l'utilisateur
//  propriétaires du fichier
function proprietaire($fichier)
{
    //  recupère l'UID du fichier
    $uid = fileowner($fichier);
    //  Recupère les informations sur l'UID
    $tabUtilisateur = posix_getpwuid($uid);

    //  recupère le GID du fichier
    $gid = filegroup($fichier);
    //  Recupère les informations sur l'GID
    $tabGroup = posix_getgrgid($gid);

    //  Renvoie les noms du groupe et de l'utilisateur
    return array(
        "utilisateur"=>$tabUtilisateur["name"],
        "groupe"=>$tabGroup["name"]
        );
}

//  Retourne la date de la dernière modification du fichier
function modificationFichier($fichier)
{
    //  Récupération de la dernière modification du fichier
    $modification = filemtime($fichier);

    //  Retourne la date de modification formatée
```

```
    return strftime("%b %d %H:%M", $modification)."\n";
}

//  Retourne les permissions sous la forme "rwx"
function uPerm($perm)
{
    switch ($perm)
    {
        case 0:
            $retPerm = "---";
            break;
        case 1:
            $retPerm = "--x";
            break;
        case 2:
            $retPerm = "-w-";
            break;
        case 3:
            $retPerm = "-wx";
            break;
        case 4:
            $retPerm = "r--";
            break;
        case 5:
            $retPerm = "r-x";
            break;
        case 6:
            $retPerm = "rw-";
            break;
        case 7:
            $retPerm = "rwx";
            break;
    }
    return $retPerm;
}

//  Retourne les permissions du fichier
function affichePermission($fichier)
{
    //  Récupère le mode du fichier et conversion en octal
    $mode = fileperms($fichier);

    //  Détermine le type du fichier
    if(($mode & 0x1000) === 0x1000)
        $type = "p";    // Port
    elseif(($mode & 0x2000) === 0x2000)
        $type = "c";    // Matériel
    elseif(($mode & 0x4000) === 0x4000)
        $type = "d";    // Répertoire
    elseif(($mode & 0x6000) === 0x6000)
        $type = "b";    // Matériel FIFO
    elseif(($mode & 0x8000) === 0x8000)
```

```
        $type = "-";    // Fichier normal
    elseif(($mode & 0xa000) === 0xA000)
        $type = "l";    // Lien symbolique
    elseif(($mode & 0xc000) === 0xC000)
        $type = "s";    // socket
    else
        $type = "u";    // Unknown

    $mode = decoct($mode);
    $perTemp = substr($mode,-3);

    $permission["utilisateur"] = uPerm(substr($perTemp,0,1));
    $permission["groupe"] = uPerm(substr($perTemp,1,1));
    $permission["tous"] = uPerm(substr($perTemp,2,1));

    return $type.$permission["utilisateur"].
        $permission["groupe"].$permission["tous"];

}

//  Retourne l'espace disque libre et total
function espaceDisque($repCourant)
{
    $Mo = pow(2, 20);
    //  Retourne une chaine de caractères indiquant l'espace disque
    // disponible sur l'espace disque total.
    $chDisque = round(disk_free_space($repCourant)/$Mo, 1)."Mo";
    $chDisque .= " / ";
    $chDisque .= round(disk_total_space($repCourant)/$Mo, 1)."Mo";

    return $chDisque;
}

// Liste et affiche le contenu du répertoire courant
function listRepertoire()
{
    // Récupération du chemin courant
    $repCourant = getcwd();

    $fichiers = explorer(".");
    // Nous ajouterons ".." (qui a été filtré par la fonction)
    if ($repCourant != "/") $fichiers = array_merge(array("./.."), $fichiers);
?>
<table border="1" width="100%">
    <tr>
        <td><font color="#cc0000">
        <?php echo $repCourant; ?>
        </font></td>
        <td>
        <?php echo espaceDisque($repCourant);?>
        </td>
    </tr>
```

```
</table>
<table border="0" width="100%">
    <form name='fliste'>   <?php // Formulaire de selection de fichier ?>
<?php
    for ($i=0; $i<count($fichiers); $i++)
    {
?>
    <tr>
        <td>
        <input type="radio" name="selection"
               value="<?php echo $fichiers[$i];?>">
        </td>
        <td>
        <?php echo affichePermission($fichiers[$i]);?>
        </td>
        <td>
        <?php
              $tabProprietaire = proprietaire($fichiers[$i]);
              echo $tabProprietaire["utilisateur"]." ";
              echo $tabProprietaire["groupe"];
 ?>
        </td>
        <td>
        <?php echo tailleFichier($fichiers[$i]);?>
        </td>
        <td>
        <?php echo modificationFichier($fichiers[$i]);?>
        </td>
        <td>
<?php
        /* Le fichier est-il un répertoire ? */
        if (is_dir($fichiers[$i]))
        {
            /*
            Le fichier est un répertoire
            On affiche alors un lien qui va nous permettre
            de visualiser le contenu de ce dossier
            */
?>
            <a href="?repertoire=<?php echo $repCourant."/".$fichiers[$i];?>">
            <?php echo basename($fichiers[$i]);?>
            </a>
        <?php
        }else{
            /*
            Le fichier n'est pas un répertoire
            On affiche simplement le nom du fichier
            */
            echo basename($fichiers[$i]);
        }
 ?>
        </td>
```

```
    </tr>
<?php
    }
?>
    </form>
</table>
<?php
/*
    Affichage des différents formulaires HTML nous
    permettant de manipuler les fichiers et répertoires.
*/
?>
<table border='0' width='100%'>
<?php /*  Copie de fichier  */?>
    <tr>
    <form method='post' action='gestionnairefichier.php' name='fcopie'>
        <td>
        <b>Copie d'un fichier</b>  
        <input type='hidden' name='repertoire'
               value='<?php echo $repCourant;?>'>
        <input type='hidden' name='idFichier'>
        nom du nouveau fichier <input type='text' name='nomCopy'>
        <input type='hidden' value='copy' name='operation'>
        <input type='button' value=' Copier '
        onclick='
        for (var i = 0; i < document.fliste.selection.length; i++) {
            if (document.fliste.selection[i].checked) {
                document.fcopie.idFichier.value =
                document.fliste.selection[i].value;
                break;
            }
        }
        document.fcopie.submit();
        '>
        </td>
    </form>
    </tr>

<?php /*  Création d'un lien  */?>
    <tr>
    <form method='post' action='gestionnairefichier.php' name='flien'>
        <td>
        <b>Création d'un lien sur un fichier</b>  
        <input type='hidden' name='repertoire'
               value='<?php echo $repCourant; ?>'>
        <input type='hidden' name='idFichier'>
        nom du lien <input type='text' name='nomLien'>
        <input type='hidden' value='link' name='operation'>
        <input type='button' value=' Créer '
        onclick='
        for (var i = 0; i < document.fliste.selection.length; i++) {
            if (document.fliste.selection[i].checked) {
```

```
                document.flien.idFichier.value =
                document.fliste.selection[i].value;
                break;
            }
        }
        document.flien.submit();
        '>
        </td>
    </form>
    </tr>

<?php /*  Création d'un lien symbolique  */?>
    <tr>
    <form method='post' name='flienSymb'>
        <td>
        <b>Création d'un lien symbolique sur un fichier</b>  
        <input type='hidden' name='repertoire'
               value='<?php echo $repCourant;?>'>
        <input type='hidden' name='idFichier'>
        nom du lien symbolique <input type='text' name='nomLienSymbolique'>
        <input type='hidden' value='symlink' name='operation'>
        <input type='button' value=' Créer '
        onclick='
        for (var i = 0; i < document.fliste.selection.length; i++) {
            if (document.fliste.selection[i].checked) {
                document.flienSymb.idFichier.value =
                document.fliste.selection[i].value;
                break;
            }
        }
        document.flienSymb.submit();
        '>
        </td>
    </form>
    </tr>

<?php /*  Renommer un fichier  */?>
    <tr>
    <form method='post' name='frenom'>
        <td>
        <b>Renommer un fichier</b>  
        <input type='hidden' name='repertoire'
               value='<?php echo $repCourant;?>'>
        <input type='hidden' name='idFichier'>
        nouveau nom du fichier <input type='text' name='nomFichier'>
        <input type='hidden' value='rename' name='operation'>
        <input type='button' value=' Renommer '
        onclick='
        for (var i = 0; i < document.fliste.selection.length; i++) {
            if (document.fliste.selection[i].checked) {
                document.frenom.idFichier.value =
                document.fliste.selection[i].value;
```

```
                break;
            }
        }
        document.frenom.submit();
        '>
        </td>
    </form>
    </tr>

<?php /*  Créer un dossier  */?>
    <tr>
    <form method='post' name='fdossier'>
        <td>
        <b>Créer un dossier</b>  
        <input type='hidden' name='repertoire'
               value='<?php echo $repCourant;?>'>
        nom du dossier <input type='text' name='nomDossier'>
        <input type='hidden' value='mkdir' name='operation'>
        <input type='button' value=' Créer '
               onclick='document.fdossier.submit();'>
        </td>
    </form>
    </tr>

<?php /*  Suppression d'un fichier  */?>
    <tr><form method='post' name='fsupprim'>
        <td>
        <b>Supprimer un fichier</b>  
        <input type='hidden' name='repertoire'
               value='<?php echo $repCourant;?>'>
        <input type='hidden' name='idFichier'>
        <input type='hidden' value='unlink' name='operation'>
        <input type='button' value=' Supprimer '
        onclick='
        for (var i = 0; i < document.fliste.selection.length; i++) {
            if (document.fliste.selection[i].checked) {
                document.fsupprim.idFichier.value =
                document.fliste.selection[i].value;
                break;
            }
        }
        document.fsupprim.submit();
        '>
        </td>
    </form></tr>
</table>
<?php
}
?>
```

Quant au script principal, il ne devra plus se contenter de permettre la navigation dans les répertoires, mais il devra également prendre en charge les différentes actions proposées (suppression, renommage, création de fichiers et de répertoires).

Listing 9.31 : gestionnaire_fichier_07.php

```
<?php
include ("gestionnaire_fichier_07_inc.php");

switch ($_POST["operation"])
{
    //  Copie de fichier
    case "copy":
        if (@copy($_POST["repertoire"]."/".$_POST["idFichier"],
                  $_POST["repertoire"]."/".$_POST["nomCopy"]))
        {
            $message = "La copie du fichier a été effectuée.";
        } else {
            $message = "Erreur pendant la copie du fichier.<br>";
        }
        break;

    //  Création d'un lien
    case "link":
        if (@link($_POST["repertoire"]."/".$_POST["idFichier"],
                  $_POST["repertoire"]."/".$_POST["nomLien"]))
        {
            $message = "La création du lien a été effectuée.";
        } else {
            $message = "Erreur pendant la création du lien.";
        }
        break;

    //  Supprimer un fichier
    case "unlink":
        if (is_dir($_POST["repertoire"]."/".$_POST["idFichier"]))
        {
            if (@rmdir($_POST["repertoire"]."/".$_POST["idFichier"]))
            {
                $message = "Le dossier a été supprimé.";
            } else {
                echo $_POST["repertoire"]."/".$_POST["idFichier"];
                $message = "Erreur, vérifiez que le dossier est vide".
                           " avant de le supprimer.";
            }
        }else{
            if (@unlink($_POST["repertoire"]."/".$_POST["idFichier"]))
            {
                $message = "Le fichier a été supprimé.";
            } else {
                $message = "Erreur pendant la suppression du fichier.";
            }
```

```
        }
        break;

    //  Création d'un lien symbolique
    case "symlink":
        if (@symlink($_POST["repertoire"]."/".$_POST["idFichier"],
                $_POST["repertoire"]."/".$_POST["nomLienSymbolique"]))
        {
            $message = "Le fichier a été supprimé.";
        } else {
            $message = "Erreur pendant la suppression du fichier.";
        }
        break;

    //  Renommer un fichier
    case "rename":
        if (@rename($_POST["repertoire"]."/".$_POST["idFichier"],
                $_POST["repertoire"]."/".$_POST["nomFichier"]))
        {
            $message = "Le fichier a été renommé.";
        } else {
            $message = "Erreur, impossible de renommer le fichier.";
        }
        break;

    //  Création d'un repertoire
    case "mkdir":
        umask(000);
        if (@mkdir($_POST["repertoire"]."/".$_POST["nomDossier"], 0760))
        {
            $message = "Le repertoire a été créé.";
        } else {
            $message = "Erreur, impossible de créer le dossier.";
        }
        break;
}

// Vérifie si "repertoire" est passé en paramètre
if ($_GET["repertoire"])
{
    if (!@chdir($_GET["repertoire"]))
    {
        $message = "Le changement de répertoire a échoué.";
    }
}

/*  Dans le cas ou les données sont envoyées par une méthode GET */
if($_GET["repertoire"])
{
    if(!@chdir($_GET["repertoire"]))
    {
        $message = "Le changement de répertoire a échoué.";
```

```
    }
}
/*  Dans le cas ou les données sont envoyées par une méthode POST */
elseif ($_POST["repertoire"])
{
    if(!@chdir($_POST["repertoire"]))
    {
        $message = "Le changement de répertoire a échoué.";
    }
}

listRepertoire();
?>
```

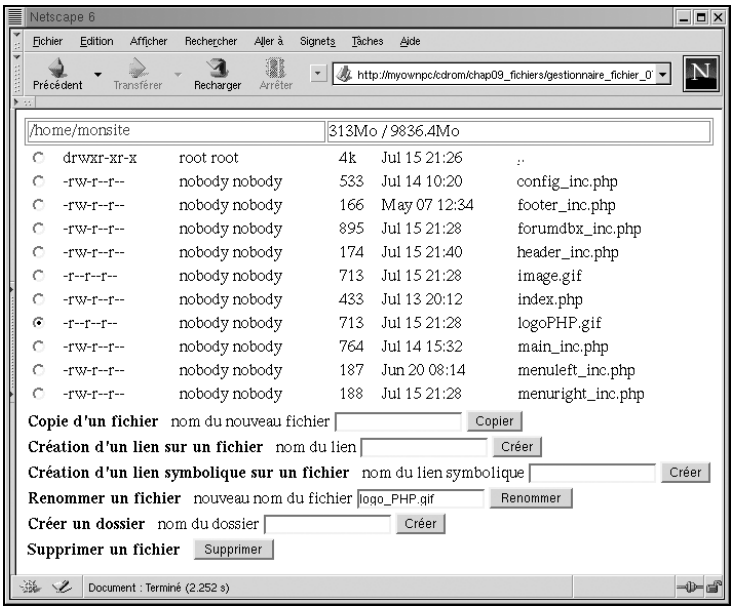


Figure 9.13 : *Le gestionnaire de fichiers*

Lecture sur un "pipe"

PHP offre la possibilité de lancer un processus depuis un script et de communiquer avec lui sur un canal de communication que l'on appelle alors *pipe* (tuyau). Pour cela, il faut utiliser l'instruction `popen()`.

popen()

Lance un processus et ouvre un *pipe* de communication.

Syntaxe `resource popen(string $commande, string $mode)`

`$commande`	Commande à exécuter.
`$mode`	Définit le mode d'ouverture. Contrairement à `fopen()`, ici le mode ne peut être qu'en lecture seule '`r`' ou en écriture seule '`w`'.
`retour`	Pointeur sur le processus créé, ou `FALSE` s'il y a une erreur.

```
<?php
$pp = popen("more fichier.txt", "r");
?>
```

Il n'est pas possible de fermer le processus à l'aide de l'instruction `fclose()`. Le langage PHP met à la disposition des développeurs la fonction `pclose()` qui ferme et libère le processus créé par `popen()`.

pclose()

Termine un processus et ferme le *pipe* associé.

Syntaxe	`boolean pclose(resource $pp)`
`$pp`	Pointeur du *pipe* tel que retourné par la fonction `popen()`.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

Voici un exemple (dans la série "pourquoi faire simple quand on peut faire compliqué".) utilisant la commande `more` pour lire le contenu d'un fichier.

```
<?php
$pp = popen("more fichier.txt", "r");
while ($chaine = fgets($pp, 255))
{
    echo $chaine;
}
pclose($pp);
?>
```

Ce qui, dans notre cas, retourne (tout bêtement) :

```
::::::::::::::
fichier.txt
::::::::::::::
voici le texte que j'ai placer dans fichier.txt
Bon, ça ne fait pas avancer le schmilblik
```

Compression

PHP propose différentes bibliothèques liées à la compression, mais la bibliothèque `Zlib` est la seule à fonctionner parfaitement (contrairement à `bzip2`), comme elle est la seule à permettre à la fois la compression et la décompression (contrairement à `zip`).

Compression Zlib

`Zlib` est la bibliothèque à la base de Gzip (GNUZip), qui est lui-même l'utilitaire de compression le plus utilisé sous UNIX. Toutes les distributions Linux fournissent cet utilitaire, ce qui est bien normal lorsqu'on sait qu'il est développé par le Projet GNU de la Free Software Foundation (**http://www.gnu.org**). Rien d'étonnant donc à trouver, dans PHP, diverses commandes permettant d'exploiter la librairie `Zlib`.

En ligne de commande, la compression est effectuée avec la commande `gzip nomArchive`. Le fichier est alors remplacé par un nouveau fichier possédant l'extension *.gz*. Attention, car Gzip ne peut en aucun cas compresser plusieurs fichiers à la fois. Si vous désirez compresser une série de fichiers, vous devez au préalable les archiver (avec la commande `tar` par exemple). La décompression s'effectue avec la commande `gzip -d nomArchive.gz`. Si le fichier n'est pas un fichier compressé au format Gzip, alors la décompression échouera et renverra une alerte.

Le langage PHP s'est donc interfacé avec la bibliothèque `Zlib`, ce qui permet d'effectuer diverses opérations courantes de compression et de décompression.

Pour information, la `Zlib` utilise, entre autres, l'algorithme de Huffman pour compresser les données. Le principe de cet algorithme est de modifier l'encodage des caractères en analysant la fréquence de répétition de ces caractères. L'algorithme les classe du plus au moins fréquent. L'unité de traitement étant ramenée au bit, chaque caractère possède alors un nouvel encodage sur un nombre de bits variable. Ainsi, le caractère le plus utilisé possède une valeur sur un petit nombre de bits et le caractère le moins utilisé est encodé sur un nombre de bits plus grand.

Une fois encodée, une donnée répétitive occupe moins de place que l'original (il y a bien compression) tandis qu'une donnée rare occupe plus de place (qui doit être largement compensée par la compression). Le tableau suivant permet de mieux comprendre ce principe.

Imaginez un texte (autre que *la Disparition* de Georges Perec qui ne contient pas de "e") qui posséderait 15 fois la lettre "w", 260 fois la lettre "e" et 510 fois la lettre "a". Habituellement, un caractère est codé sur 8 bits. Nous pouvons imaginer qu'avec la compression `Zlib`, le caractère "w" (peu représenté) occupe 12 bits, le caractère "e" (bien représenté) 6 bits et, enfin, le caractère "a" (très présent) 2 bits. Dans ce cas le texte ne fera pas (15+260+510)*8 = 6280 mais 15*12+260*6+510*2=2760, ce qui constitue bien une compression du texte.

Cette technique d'encodage est à la base des compressions `Jpeg`, `Tiff` et aussi, donc, de la compression `Zlib`.

Installation

Sous Windows

Depuis PHP 4.3.0 vous n'avez rien à faire pour disposer de ces fonctions. Si toutefois vous souhaitez utiliser une version antérieure alors vérifiez que vous avez activé l'extension *php_zlib.dll*.

Sous Linux

Pour installer le module `Zlib`, vous devez, dans un premier temps, récupérer les sources de la bibliothèque disponible sur le site (en anglais) **http://www.gzip.org/zlib/** (ou sur le CD-ROM fourni).

Vous pouvez alors copier l'archive dans un répertoire quelconque (*/usr/local/lib*, par exemple) et commencer sa décompression.

```
# gunzip zlib-1.1.4.tar.gz
# tar xvf zlib-1.1.4.tar
```

Vous utiliserez la méthode classique de compilation et d'installation,

```
# cd zlib-1.1.4
# ./configure
# make
# make install
```

ce qui aura pour effet d'ajouter un fichier *libz.a* dans le répertoire */usr/local/lib*.

Distributions Linux

Comme vous l'aurez remarqué, la procédure d'installation décrite ici fait elle-même appel à Gzip. Ceci ne devrait pas constituer un problème, puisque Gzip est généralement installé avec Linux. Sachez toutefois que l'archive Gzip est disponible sous d'autres formats de compression.

De même, il n'est pas forcément nécessaire de recompiler cette bibliothèque comme indiqué précédemment, puisque les distributions Linux incluent généralement des "paquetages" de Gzip pour le développement (y sont inclus les en-têtes et la bibliothèque compilée).

Ainsi, sous Mandrake 8.1 (par exemple), la procédure d'installation peut être remplacée par :

```
# rpm -U zlib1-1.1.3-16.1mdk.i586.rpm
# rpm -U zlib1-devel-1.1.3-16.1mdk.i586.rpm
```

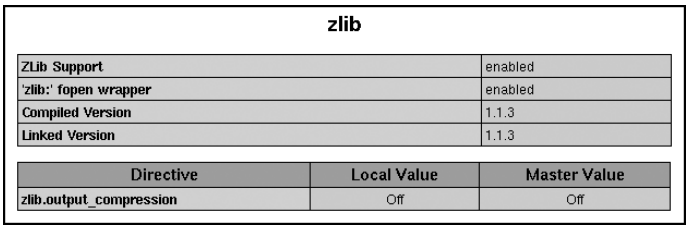
Quoi qu'il en soit, vous devez nécessairement être équipé d'une version supérieure ou égale à 1.0.9.

Vous devez alors recompiler PHP avec l'option *--with-zlib=/usr/local*.

Vous pouvez vous reporter au chapitre "Prise en main" pour plus de détails sur la compilation de PHP.

Vérification

Vérifiez que le module `Zlib` est bien installé en appelant une page contenant `<?php phpinfo(); ?>`.



zlib

ZLib Support	enabled
'zlib:' fopen wrapper	enabled
Compiled Version	1.1.3
Linked Version	1.1.3

Directive	Local Value	Master Value
zlib.output_compression	Off	Off

Figure 9.14 : *Le phpinfo vous informe de l'installation du module zip.*

Ouvrir et fermer un fichier Gzip

Le module `Zlib` fonctionne à peu près comme les fonctions du système de fichiers. Vous allez retrouver la plupart de ces fonctions. De plus, depuis la version 4.0.4 de PHP, il est possible d'ouvrir un fichier compressé au format *gzip* depuis un appel classique à `fopen()`. Il suffit de faire précéder le chemin du fichier à ouvrir par "`compress.zlib://`" (`zlib:` pour PHP<4.3.0), comme le montre l'exemple ci-dessous.

```
<?php
$fp = fopen("compress.zlib://monFichier.txt.gz","r");
fclose($fp);
?>
```

L'ouverture d'un fichier compressé avec Gzip se fait à l'aide de l'instruction `gzopen()`. Comme pour la fonction `fopen()`, le développeur doit indiquer le mode d'ouverture (écriture seule, lecture seule ou lecture/écriture), et peut également préciser un niveau de compression ainsi qu'une stratégie de compression.

Rendez-vous au début de ce chapitre, à la description de la fonction `fopen()`, pour connaître les différents modes possibles et leurs particularités.

gzopen()

Ouverture d'un fichier compressé au format *gzip*.

Syntaxe	`resource gzopen(string $nomFichier, string $mode [, boolean $cheminInclude])`
`$nomFichier`	Chemin du fichier.
`$mode`	Définit le mode d'ouverture du fichier : "`r`" pour la lecture seule. "`r+`" pour une lecture et une écriture. "`w`" pour une écriture avec écrasement des données. "`w+`" pour un mode en lecture et écriture avec écrasement des données. "`a`" pour l'ouverture en ajout de données. "`a+`" pour l'ouverture du fichier en mode lecture et écriture par ajout.

	Vous pouvez ajouter le niveau de compression entre "0"(aucune compression) et "9"(compression maximale) ; par défaut le niveau est de "6". Vous pouvez spécifier une stratégie : "f" pour des données filtrées. "h" pour l'utilisation de l'algorithme de Huffman uniquement.
$cheminInclude	TRUE si le fichier doit être recherché dans les répertoires précisés par include_path, FALSE (valeur par défaut) sinon.
retour	Pointeur sur le fichier, ou FALSE en cas d'erreur.

Ainsi le mode "a+b6h" désigne une ouverture en lecture et écriture par ajout d'un fichier binaire avec un niveau de compression de 6 et l'utilisation exclusive de l'algorithme de Huffman.

De même, le mode "wb9f" indique une ouverture en mode écriture d'un fichier binaire d'un niveau de compression de 9 avec une stratégie filtrée.

La fermeture du fichier est effectuée à l'aide de la fonction gzclose().

gzclose()

Ferme un fichier préalablement ouvert avec l'instruction gzopen().

Syntaxe	boolean gzclose(resource $gzFichier)
$gzFichier	Pointeur vers le fichier tel que retourné par la fonction gzopen().
retour	TRUE si le fichier a été fermé, FALSE dans le contraire.

```
<?php
$gzFichier = gzopen("monfichier.txt.gz","w");
//  Traitement
gzclose($gzFichier);
?>
```

Le petit programme ci-dessous est équivalent à l'exemple précédent :

```
<?php
$gzFichier = fopen("zlib:monfichier.txt.gz","w");
//  Traitement
fclose($gzFichier);
?>
```

Ouverture des fichiers non compressés
La fonction gzopen() peut également ouvrir des fichiers non compressés. Dans ce cas, les lecture et écriture se feront sans compression ni décompression.

Écrire dans un fichier

L'instruction `gzwrite()` permet d'écrire une chaîne de caractères dans un fichier ouvert préalablement à l'aide de l'instruction `gzopen()`. L'appel à `gzwrite()` écrit donc une chaîne à la position courante du pointeur.

gzwrite()

Écriture d'une chaîne de caractères dans un fichier.

Syntaxe	`int gzwrite(resource $gzFichier, string $chaine [, int $longueur])`
`$gzFichier`	Pointeur vers le fichier ouvert avec l'instruction `gzopen()`.
`$chaine`	Chaîne de caractères à écrire dans le fichier.
`$longueur`	Paramètre optionnel indiquant combien de caractères doivent être écrits. S'il est omis, la chaîne complète est écrite dans le fichier.
`retour`	Nombre de caractères qui ont été écrits dans le fichier. En cas d'erreur, l'instruction retourne `FALSE`.

Tout comme l'instruction `fwrite()`, la fonction `gzwrite()` possède un alias appelé `gzputs()`.

Listing 9.32 : gzwrite.php

```
<?php
$maChaine = "\"Je suis un phénomène, je suis un magicien\n".
            "Un tuyau parcouru d'informations majeures\n".
            "Mais y'a pas que des douceurs qui passent à l'intérieur\"\n".
            "Noir Désir, Son style\n";

$gzFichier = gzopen("monfichier.txt.gz","w");
gzwrite($gzFichier, $maChaine);
gzclose($gzFichier);
?>
```

Vous pouvez observer qu'un nouveau fichier a fait son apparition dans le répertoire courant. Dézippez-le avec la commande `gzip -d monfichier.txt.gz` et observez que votre texte s'y trouve bel et bien (ou affichez directement son contenu avec `zcat monfichier.txt.gz`). Vous pouvez utiliser WinZip ou tout autre outil de décompression supportant le format *gz* disponible sous Windows.

Lire des informations dans un fichier

Tout comme pour un fichier classique, la lecture des données contenues dans un fichier peut se faire octet par octet, bloc de N octets par bloc de N octets, ligne par ligne ou encore d'un bloc.

Ceci est assuré par les différentes fonctions que sont `gzread()`, `gzgets()`, `gzgetss()`, `gzgetc()` et deux instructions particulières que sont `gzfile()` et `readgzfile()`.

Lecture octet par octet

La fonction `getc()` possède son équivalence pour traiter les fichiers au format *gzip*. Cette instruction est `gzgetc()`.

gzgetc()

Retourne le caractère se trouvant à la position courante du pointeur de lecture.

Syntaxe	`string gzgetc(resource $gzFichier)`
`$gzFichier`	Pointeur vers le fichier tel que retourné par `gzopen()`.
`retour`	Caractère à la position courante du pointeur de lecture ; `FALSE` en cas d'erreur ou lorsque le pointeur de lecture se trouve à la fin du fichier.

Lecture N octets par N octets

Également semblable à la lecture dans un fichier texte ou binaire classique. La fonction `gzread()` permet la lecture par paquet d'octets.

gzread()

Lecture des données décompressées contenues dans un fichier au format *gzip*. La lecture s'effectue par paquet d'octets.

Syntaxe	`string gzread(resource $gzFichier, int $longueur)`
`$gzFichier`	Pointeur vers le fichier ouvert tel que retourné par `gzopen()`.
`$longueur`	Taille en octets des données à lire. Cette taille est indiquée pour des données une fois décompressées. La lecture se termine lorsque le fichier a été entièrement lu, ou lorsque le nombre d'octets spécifié a été retourné.
`retour`	La chaîne de caractères décompressée ; `FALSE` en cas d'erreur ou lorsque le pointeur de lecture se trouve à la fin du fichier.

L'exemple qui suit :

Listing 9.33 : gzread.php

```
<?php
$gzFichier = gzopen("monfichier.txt.gz","r");

// Affiche le contenu 50 caractères par 50 caractères
while ($maChaine = gzread($gzFichier, 50)){
    echo $maChaine."<br />";
}
gzclose($gzFichier);
?>
```

offrira donc le résultat suivant :

```
"Je suis un phénomène, je suis un magicien Un tuya
u parcouru d'informations majeures Mais y'a pas qu
e des douceurs qui passent à l'intérieur" Noir Dés
ir, Son style
```

Lecture ligne par ligne

Comme le montre l'exemple précédent, la fonction `gzread()` n'est pas adaptée à la lecture ligne par ligne. L'instruction `gzgets()` permet de retourner, au contraire, une ligne entre chaque lecture.

gzgets()

Retourne une ligne décompressée depuis le fichier au format *gzip*.

Syntaxe	`string gzgets(resource $gzFichier, int $longueur)`
`$gzFichier`	Pointeur sur un fichier tel que retourné par `gzopen()`.
`$longueur`	Taille maximale (en octets) des données à lire. Cette taille est indiquée pour des données une fois décompressées. La lecture se termine lorsque la fin de la ligne a été rencontrée, ou lorsque le nombre d'octets spécifié a été retourné.
`retour`	Une chaîne de caractères décompressée ; `FALSE` en cas d'erreur ou lorsque le pointeur de lecture se trouve à la fin du fichier.

Listing 9.34 : gzgets.php

```
<?php
$gzFichier = gzopen("monfichier.txt.gz","r");
//  Affiche une ligne et un retour à la ligne
//  entre chaque affichage
while ($maChaine = gzgets($gzFichier, 255)){
    echo $maChaine."<br />";
}
gzclose($gzFichier);
?>
```

Cet exemple retournera bien le résultat escompté, à savoir :

```
"Je suis un phénomène, je suis un magicien
Un tuyau parcouru d'informations majeures
Mais y'a pas que des douceurs qui passent à l'intérieur"
Noir Désir, Son style
```

Lecture d'un bloc

Tout comme l'instruction `file()`, vous pouvez retourner directement le fichier décompressé dans un tableau contenant chaque ligne. Pour cela nous utiliserons la fonction `gzfile()`.

gzfile()

Retourne le fichier décompressé dans un tableau.

Syntaxe	`array gzfile(string $nomFichier [, boolean $cheminInclude])`
`$nomFichier`	Chemin du fichier.
`$cheminInclude`	`TRUE` si le fichier doit être recherché dans les répertoires précisés dans `include_path`, `FALSE` (par défaut) sinon.
`retour`	Tableau contenant chaque ligne du fichier décompressé, ou `FALSE` si une erreur est survenue.

Listing 9.35 : gzfile.php

```
<?php
$gzTableau = gzfile("monfichier.txt.gz");
//  Regroupe tous les éléments du tableau
//  dans une chaîne de caractères
$chaine = implode("<br />", $gzTableau);
//  Affiche la chaîne de caractères
echo $chaine;
?>
```

L'exemple précédent affiche simplement le contenu du fichier décompressé (puisque le tableau est immédiatement converti en une chaîne de caractères en insérant des retours à la ligne entre chaque ligne).

Plus simplement, le langage PHP possède la fonction `readgzfile()`. Cette instruction décompresse et affiche le fichier.

readgzfile()

Affiche le contenu décompressé du fichier.

Syntaxe	`int readgzfile(string $nomFichier [, boolean $cheminInclude])`
`$nomFichier`	Chemin du fichier.
`$cheminInclude`	`TRUE` si le fichier doit être recherché dans les répertoires précisés dans `include_path`, `FALSE` (par défaut) sinon.
`retour`	Nombre d'octets décompressés qui ont été lus, ou `FALSE` en cas d'erreur.

Listing 9.36 : readgzfile.php

```
<?php
$nombreOctets = readgzfile("monfichier.txt.gz");
echo "<br />";
echo "Nombre d'octets lu : ".$nombreOctets;
?>
```

```
"Je suis un phénomène, je suis un magicien Un tuyau parcouru d'informations
✂ majeures Mais y'a pas que des douceurs qui passent à l'intérieur" Noir Désir,
✂ Son style
Nombre d'octets lu : 163
```

Dans ce cas, parce que HTML n'interprète pas les retours chariots '`\n`' tout le texte apparaîtra sur une ligne unique (dans la limite de la taille de la fenêtre).

Lorsque le fichier a déjà été ouvert, il est possible d'afficher le reste d'un fichier depuis la position courante du pointeur de lecture. L'instruction `gzpassthru()` est la version pour les fichiers compressés au format *gzip*, de l'instruction `fpassthru()`.

gzpassthru()

Retourne sur l'écran le contenu du fichier depuis la position courante du pointeur de lecture.

Syntaxe	`int gzpassthru(resource $gzFichier)`
`$gzFichier`	Pointeur du fichier tel que retourné par `gzopen()`.
`retour`	Nombre d'octets décompressés qui ont été affichés, ou `FALSE` en cas d'erreur.

Listing 9.37 : gzpassthru.php

```
<?php
$gzFichier = gzopen("monfichier.txt.gz","r");
//  Affiche les 30 premiers caractères et un retour à la ligne.
echo gzread($gzFichier, 30);
echo "<br /><br />";
//  Affiche le reste du fichier décompressé sur l'écran.
$nbOctets = gzpassthru($gzFichier);
echo "<br />";
echo "Nombre d'octets affichés par gzpassthru() : ".$nbOctets;
?>
```

```
"Je suis un phénomène, je suis
un magicien Un tuyau parcouru d'informations majeures Mais y'a pas que des
✂ douceurs qui passent à l'intérieur" Noir Désir, Son style
Nombre d'octets affiché par gzpassthru() : 133
```

Remarquez qu'il est inutile de fermer le fichier par l'instruction `gzclose()`. En effet, comme pour la fonction `fpassthru()`, le fichier est automatiquement fermé.

Lecture et filtrage HTML

Vous pouvez également filtrer les fichiers HTML compressés. Pour ce faire, vous utiliserez l'instruction `gzgetss()`. Cette fonction retourne le fichier ligne par ligne en enlevant les balises HTML, à l'exception de celles que vous aurez spécifiées.

gzgetss()

Retourne une ligne d'un fichier compressé sans les balises HTML (sauf celles que l'on souhaite conserver).

Syntaxe	`string gzgetss(resource $gzFichier, int $longueur [, string $tagsAutorise])`
`$gzFichier`	Pointeur du fichier tel que retourné par `gzopen()`.
`$longueur`	Taille maximale (en octets) des données à lire. Cette taille est indiquée pour des données une fois décompressées. La lecture se termine lorsque le fichier a été entièrement lu ou lorsque le nombre d'octets spécifié a été retourné.
`$tagsAutorise`	Les balises qui ne doivent pas être supprimées. Vous pouvez spécifier plusieurs tags en les séparant par le caractère "|". Ex. : " |<b>| ".
`retour`	Chaîne contenant les données lues de la ligne courante sans les balises HTML ; `FALSE` en cas d'erreur ou si le pointeur se trouve à la fin du fichier.

```
<?php
$fp = gzopen("monFichier.html.gz","r");
while ($ligne = gzgetss($fp, 255, "<br>|<center>"))
{
    echo $ligne;
}
gzclose($fp);
?>
```

L'exemple ci-dessus affiche le fichier HTML décompressé en enlevant toutes les balises sauf celles qui correspondent à `<br>` et `<center>`.

Positionner le pointeur de lecture/écriture

Tout comme pour la lecture et l'écriture des données dans un fichier non compressé, vous pouvez déplacer le pointeur de lecture/écriture. Ce pointeur est positionné par rapport au fichier décompressé. Les quatre fonctions `feof()`, `ftell()`, `fseek()` ou `rewind()` ont donc leurs équivalents pour les fichiers au format *gzip*. Ces fonctions sont `gzeof()`, `gztell()`, `gzseek()`, `gzrewind()`.

gzeof()

Teste la fin du fichier décompressé.

Syntaxe	`boolean gzeof(resource $gzFichier)`
`$gzFichier`	Pointeur du fichier tel que retourné par `gzopen()`.
`retour`	`TRUE` si le pointeur est à la fin du fichier, ou `FALSE` dans le cas contraire.

Listing 9.38 : gzeof.php

```
<?php
//  Ouverture du fichier
$gzFichier = gzopen("monfichier.txt.gz","r");
//  Initialise la variable $i à "0"
$i = 0;
//  Lit le fichier octet par octet jusqu'à
//  la fin du fichier
while (!gzeof($gzFichier))
{
    //  Lit un caractère et déplace le pointeur
    //  de lecture d'un octet
    gzgetc($gzFichier);
    //  Indente la variable $i
    $i++;
}
echo "Le fichier possède ".($i-1)." octets.";
gzclose($gzFichier);
?>
```

retournera :

```
Le fichier possède 162 octets.
```

Cet exemple montre comment fonctionne la fonction `gzeof()` et n'a d'autre valeur que pédagogique. La technique pour déplacer le pointeur de lecture/écriture n'est pas efficace. `gzgetc()`est à utiliser uniquement pour lire un octet. Pour déplacer un pointeur, utilisez l'instruction `gzseek()` décrite plus loin.

Pour connaître la position courante du pointeur de lecture/écriture dans un fichier, vous pouvez utiliser l'instruction `gztell()`.

gztell()

Retourne la position d'un pointeur de lecture dans le fichier décompressé.

Syntaxe	`int gztell(resource $gzFichier)`

`$gzFichier`	Pointeur du fichier tel que retourné par `gzopen()`.
`retour`	Position courante du pointeur de lecture, et `FALSE` si une erreur survient.

Le déplacement du pointeur de lecture/écriture est effectué avec l'instruction `gzseek()`. Cette fonction fixe la position du pointeur par rapport au début du fichier. Notez que, contrairement à l'instruction `fseek()`, vous n'avez pas la possibilité de fixer une origine pour le déplacement.

gzseek()

Fixe une nouvelle position au pointeur de lecture/écriture par rapport au début du fichier.

Syntaxe	`int gzseek(resource $gzFichier, int $nbOctets)`
`$gzFichier`	Pointeur du fichier tel que retourné par `gzopen()`.
`$nbOctets`	Nombre d'octets dont le pointeur doit être déplacé par rapport au début du fichier. Notez que si vous déplacez le pointeur après la fin du fichier, cela ne renvoie pas d'erreur.
`retour`	Nouvelle position du pointeur, ou `-1` en cas d'erreur.

L'instruction `gzrewind()` permet de repositionner le pointeur de lecture/écriture au début du fichier.

gzrewind()

Repositionne le pointeur au début du fichier.

Syntaxe	`boolean gzrewind (resource $gzFichier)`
`$gzFichier`	Pointeur du fichier tel que retourné par `gzopen()`.
`retour`	`TRUE` si le pointeur a bien été positionné au début du fichier, `FALSE` dans le cas contraire.

Listing 9.39 : gztell.php

```
<?php
//  Ouverture du fichier
$gzFichier = gzopen("monfichier.txt.gz","r+");

//  Déplace le pointeur de 8 caractères.
gzseek($gzFichier, 8);
//  Affiche la position du pointeur.
echo "Nouvelle position du pointeur de lecture/écriture : ";
echo gztell($gzFichier);
echo "<br />";
```

```
//  Déplace le pointeur de 50 caractères.
gzseek($gzFichier, 50);
//  Affiche la position du pointeur.
echo "Nouvelle position du pointeur de lecture/écriture : ";
echo gztell($gzFichier);
echo "<br />";

//  Réinitialise le pointeur au début du fichier.
gzrewind($gzFichier);
//  Affiche la position du pointeur.
echo "Nouvelle position du pointeur de lecture/écriture : ";
echo gztell($gzFichier);
echo "<br />";
gzclose($gzFichier);
?>
```

ce qui affichera :

```
Nouvelle position du pointeur de lecture/écriture : 8
Nouvelle position du pointeur de lecture/écriture : 50
Nouvelle position du pointeur de lecture/écriture : 0
```

Compression et décompression d'une chaîne de caractères

Le module `Zlib` permet également de compresser directement une chaîne de caractères sans nécessairement passer par un fichier.

Avec gzcompress

gzcompress()

Compresse une chaîne de caractères en utilisant l'algorithme de la librairie `zziplib`. Mais attention, il est impossible d'utiliser l'outil *Gzip* pour décompresser les fichiers ainsi créés. Utilisez `gzencode()` pour obtenir des archives compatibles.

Syntaxe	`string gzcompress(string $chaine [, int $facteur])`
`$chaine`	Chaîne contenant les données à compresser.
`$facteur`	Indique le niveau de compression entre `0` (pour une compression nulle) et `9` (pour une compression maximale). Plus le facteur de compression est important, plus il y a utilisation des ressources. Par défaut, le facteur est de `6`.
`retour`	Données compressées.

Listing 9.40 : gzdeflate.php

```
<?php
$maChaine = "\"Je suis un phénomène, je suis un magicien\n".
```

```
            "Un tuyau parcouru d'informations majeures\n".
            "Mais y'a pas que des douceurs qui passent à l'intérieur\"\n".
            "Noir Désir, Son style\n";
$codeHTML = "<html><head><title>".
            "Test de compression avec compress".
            "</head></title><body>".
            nl2br($machaine);

if (strpos($_SERVER["HTTP_ACCEPT_ENCODING"], "compress") !== FALSE) {
    $codeHTML .= "<br /><center><i>".
        "Cette page a été compressée par le serveur<br />".
        "et admirablement bien décompressée par votre".
        " navigateur</i></center></body></html>";
    header("Content-Encoding: compress");
    echo gzcompress($codeHTML);
} else {
     $codeHTML .= "<br /><center><i>".
        "Désolé, votre navigateur ne supporte pas ce type de compression".
        "<br /> Cette page n'a donc pas été transférée compressée".
        "</i></center></body></html>";
    echo $codeHTML;
}
?>
```

Vous pouvez vous reporter à l'annexe "Les en-têtes HTTP" et à la section "Mise en cache avant émission des données pour plus d'informations" sur l'envoi (automatique) de données compressées.

La décompression pourra s'effectuer avec :

gzuncompress()

Décompresse des données compressées avec la fonction `gzcompress()`.

Syntaxe	`string gzuncompress (string $chaine [, int $longueur])`
`$chaine`	Chaîne contenant les données compressées.
`$longueur`	Longueur maximale des données décompressées. La fonction retourne `FALSE` si la taille de la chaîne de retour est plus importante.
`retour`	Retourne la chaîne décompressée, ou `FALSE` si l'instruction rencontre une erreur dans le traitement. Si la chaîne de retour est 256 fois supérieure à `$chaine`, la fonction renverra `FALSE`.

Avec gzdeflate

Un autre algorithme de compression peut être utilisé pour compresser les données. Deflate est un format de compression particulièrement bien adapté aux chaînes de caractères. Les spécifications de ce format peuvent être trouvées à l'adresse suivante : **http://www.ietf.org/rfc/rfc1951.txt**.

Pour compresser une chaîne en utilisant ce format, vous utiliserez l'instruction `gzdeflate()`.

gzdeflate()

Compresse une chaîne de caractères.

Syntaxe	`string gzdeflate(string $chaine [, int $facteur])`
`$chaine`	Chaîne contenant les données à compresser.
`$facteur`	Indique le niveau de compression entre `0` (pour une compression nulle) et `9` (pour une compression maximale). Plus le facteur de compression est important, plus le traitement est lent. Par défaut, le facteur est de `6`.
`retour`	Données compressées.

Listing 9.41 : gzcompress.php

```
<?php
$maChaine = "\"Je suis un phénomène, je suis un magicien\n".
            "Un tuyau parcouru d'informations majeures\n".
            "Mais y'a pas que des douceurs qui passent à l'intérieur\"\n".
            "Noir Désir, Son style\n";
$codeHTML = "<html><head><title>".
            "Test de compression avec deflate".
            "</head></title><body>".
            nl2br($machaine);

if (strpos($_SERVER["HTTP_ACCEPT_ENCODING"], "deflate") !== FALSE) {
    $codeHTML .= "<br /><center><i>".
        "Cette page a été compressée par le serveur<br />".
        "et admirablement bien décompressée par votre".
        " navigateur</i></center></body></html>";
    header("Content-Encoding: deflate");
    echo gzdeflate($codeHTML);
} else {
     $codeHTML .= "<br /><center><i>".
        "Désolé, votre navigateur ne supporte pas ce type de compression".
        "<br /> Cette page n'a donc pas été transférée compressée".
        "</i></center></body></html>";
    echo $codeHTML;
}
?>
```

Une chaîne compressée avec `gzdeflate()` pourra être restituée par :

gzinflate()

Décompresse les données compressées avec la fonction `gzdeflate()`.

Syntaxe	`string gzinflate(string $chaine [, int $longueur])`
`$chaine`	Chaîne contenant les données compressées.
`$longueur`	Longueur maximale des données décompressées. La fonction retourne `FALSE` si la taille de la chaîne de retour est plus importante.
`retour`	Retourne la chaîne décompressée, ou `FALSE` si l'instruction rencontre une erreur dans le traitement. Si la chaîne de retour est 256 fois supérieure à `$chaine`, la fonction renverra `FALSE`.

Avec gzencode

Pour compresser une chaîne dans un format compatible avec *Gzip*, vous devrez utiliser la fonction `gzencode()`.

gzencode()

Compresse une chaîne en utilisant le même format de compression que la commande `gzip`.

Syntaxe	`string gzencode(string $chaine [, int $facteur [, int $modeEncodage]])`
`$chaine`	Chaîne contenant les données à compresser.
`$facteur`	Indique le niveau de compression entre `0` (pour une compression nulle) et `9` (pour une compression maximale). Plus le facteur de compression est important, plus le traitement est lent. Par défaut, le facteur est de `6`.
`$modeEncodage`	Au choix : `FORCE_DEFLATE` si la compression doit inclure les en-têtes de la `Zlib`. `FORCE_GZIP` (par défaut) sinon.
`retour`	Données compressées.

Listing 9.42 : gzencode.php

```
<?php
$maChaine = "\"Je suis un phénomène, je suis un magicien\n".
            "Un tuyau parcouru d'informations majeures\n".
            "Mais y'a pas que des douceurs qui passent à l'intérieur\"\n".
            "Noir Désir, Son style\n";
$codeHTML = "<html><head><title>".
```

```
            "Test de compression avec gzip".
            "</head></title><body>".
            nl2br($machaine);

if (strpos($_SERVER["HTTP_ACCEPT_ENCODING"], "gzip") !== FALSE) {
    $codeHTML .= "<br /><center><i>".
        "Cette page a été compressée par le serveur<br />".
        "et admirablement bien décompressée par votre".
        " navigateur</i></center></body></html>";
    header("Content-Encoding: gzip");
    echo gzencode($codeHTML);
} else {
     $codeHTML .= "<br /><center><i>".
        "Désolé, votre navigateur ne supporte pas ce type de compression".
        "<br /> Cette page n'a donc pas été transférée compressée".
        "</i></center></body></html>";
    echo $codeHTML;
}
?>
```

Un fichier est à présent sur le disque, dans le répertoire courant. Vous pouvez visualiser son contenu (si vous avez accès à la console de la machine) en utilisant soit directement `zcat debian.txt.gz` soit avec `gzip -d debian.txt.gz` suivi de `more debian.txt`. Vous constatez alors que vous retrouvez bel et bien votre fichier original.

Il n'existe toutefois pas de fonction permettant de décompresser directement une telle chaîne de caractères ; vous devez donc l'utiliser directement dans une commande réclamant une chaîne compressée ou la stocker dans un fichier pour une utilisation ultérieure (décompression et lecture avec les fonctions décrites tout au long de ce chapitre).

Cependant, il est tout de même possible de s'en sortir grâce à la fonction `gzinflate()`, en prenant soin toutefois de supprimer les dix premiers caractères de la chaîne compressée.

```
$chaine = gzinflate(substr($chaineCompressee,10));
```

Checksum

Il est très simple de calculer un *checksum* basé sur le md5 d'un fichier grâce à la fonction `md5_file()`.

md5_file()

Permet de calculer le md5 d'un fichier (chaîne hexadécimale de 32 caractères).

Syntaxe	`string md5_file(string $nomFichier)`
`$nomFichier`	Fichier texte dont on veut calculer le md5.
`retour`	md5 du fichier.

9.3. Les streams ou les flux

Les streams ont été introduit avec la version 4.3 de PHP. L'idée est de généraliser ou plutôt de s'absoudre de l'objet qui doit être atteint (système de fichier, fichiers compressés, sockets, connexions ftp ou http, etc…). Ces fonctions se comportent donc comme une interface commune permettant d'accéder, via un protocole défini, aux données qui doivent être traitées.

Installation

Il n'y a aucune manipulation particulière à effectuer pour utiliser les streams. En effet, ces fonctions sont directement utilisables depuis PHP 4.3 et supérieur.

Les gestionnaires disponibles

Les gestionnaires sont des pilotes permettant de lire un flux de données en utilisant un protocole donné. S'il est possible d'ajouter des gestionnaires à l'aide de fonctions que l'on verra dans la suite de ce chapitre, plusieurs sont disponibles dès l'installation de PHP.

Il est possible de lister ces différents gestionnaires à l'aide de la fonction `stream_get_wrappers()`.

---

stream_get_wrappers()

Permet de récupérer, dans un tableau, la liste des gestionnaires disponibles.

Syntaxe	`array stream_get_wrappers(void)`
`retour`	Tableau indexé comprenant la liste de tous les gestionnaires disponibles sur le système.

```
<?php
        print_r(stream_get_wrappers());
?>
```

Le programme ci-dessus donne le résultat suivant :

```
Array
(
   [0] => php
   [1] => file
   [2] => http
   [3] => ftp
   [4] => compress.zlib
)
```

Tableau 9.4 : Liste des divers protocoles supportés

Nom du gestionnaire	Description
php	Permet l'usage des entrées/sorties. Par exemple `php://output` permet simplement de remplacer la fonction `print()` et affiche des données.
file	Gestionnaire par défaut lors de l'utilisation des fonctions `fopen()`, `fwrite()`, etc. Ce protocole permet d'accéder au système de fichiers.
http	Gestionnaire permettant d'accéder aux données se trouvant sur un serveur web via le protocole http.
ftp	Gestionnaire permettant d'accéder aux données disponibles sur un serveur FTP.
compress.zlib	Gestionnaire permettant d'accéder aux données compressées via la librairie zlib. Remplace donc les fonctions `gzopen()`, `gzclose()`, etc. en permettant l'usage des fonctions de lecture de fichiers `fopen()`, `fread()`, etc.

Si le support SSL a été activé alors vous aurez également le support des protocoles https et ftps.

Le gestionnaire file

Le gestionnaire file est optionnel. En effet, lorsqu'il n'est pas spécifié, c'est c'est celui-ci qui est utilisé par défaut.

```
<?php
$fp = fopen("file:///tmp/file.txt","w")
    or die("Impossible de créer le fichier.");
fwrite($fp, "Exemple de création d'un fichier à l'aide du gestionnaire file");
fclose($fp);

readfile("file:///tmp/file.txt");
?>
```

L'exemple précédent crée un fichier dans le répertoire /tmp (sous UNIX/Linux, indiquer C:\\chemin pour un système Windows) et lit celui-ci ensuite pour l'afficher à l'écran.

Le gestionnaire http://

La lecture des fichiers à distance est utilisable simplement en indiquant l'url du fichier.

```
<?php
$ptFlux = fopen("http://www.php.net", "r");
while ($ligne=fread($ptFlux, 8192)) {
    print($ligne);
}
fclose($ptFlux);
?>
```

Pour passer une authentification il faut indiquer l'identifiant et le mot de passe de la façon suivante :

```
<?php
$ptFlux = fopen("http://identifiant:motdepasse@www.php.net", "r");
while ($ligne=fread($ptFlux, 8192)) {
    print($ligne);
}
fclose($ptFlux);
?>
```

Le gestionnaire FTP

La lecture des fichiers en utilisant le protocole FTP se fait en indiquant l'url du fichier de la façon suivante :

```
<?php
$fp = fopen("ftp://identifiant:motdepasse@ftp.ouvaton.org/html/test.txt","w")
     or die("Impossible de créer le fichier.");
fwrite($fp, "Exemple de création d'un fichier à l'aide du gestionnaire file");
fclose($fp);

readfile("ftp://identifiant:motdepasse@ftp.ouvaton.org/html/test.txt");
?>
```

Notez qu'il est donc possible d'accédez en écriture sur un serveur FTP ce qui peut-être très pratique pour créer des fichiers à distance.

Le gestionnaire compress.zlib

Remplacer avantageusement l'utilisation des fonctions `gzread`, `gzopen`, `gzclose()`en utilisant le wrapper `compress.zlib`. En spécifiant ce gestionnaire, cela permet l'usage des fonctions d'accès aux fichiers classiques : `fopen()`, `fread()`, `fwrite()`, `fclose()`.

```
<?php
// Création du fichier compressé gzip.filtre.txt
$fp = fopen("compress.zlib://gzip.filtre.txt","w")
    or die("Impossible de créer le fichier.");
fwrite($fp, "Michael Moore :\n");
fwrite($fp, "Fahrenheit 9/11 (2003)\n");
fwrite($fp, "Bowling for Columbine (2002)\n");
fwrite($fp, "Last party 2000 (2001)\n");
fwrite($fp, "Le Bon numéro (Lucky numbers) (2000)\n");
fwrite($fp, "The Big One (1998)\n");
fwrite($fp, "Canadian bacon (1995)\n");
fwrite($fp, "Roger et moi (Roger and me) (1989)");
fclose($fp);

// Affichage du fichier décompressé
readfile("compress.zlib://gzip.filtre.txt");
?>
```

Ce qui affiche le résultat suivant :

Michael Moore :
Fahrenheit 9/11 (2003)
Bowling for Columbine (2002)
Last party 2000 (2001)
Le Bon numéro (Lucky numbers) (2000)
The Big One (1998)
Canadian bacon (1995)
Roger et moi (Roger and me) (1989)

Le gestionnaire PHP

Voici divers exemple d'utilisation du protocole PHP

Dans un premier temps voici comment utiliser `fopen()` pour écrire dans le buffer de sortie. Le résultat est identique à l'appel d'une fonction `print()`.

```
<?php
$ptFlux = fopen("php://output", "w");
fwrite($ptFlux, "Voici un premier message\n");
fwrite($ptFlux, "Voici un deuxième message");
fclose($ptFlux);
?>
```

Il est simple de récupérer les données envoyées via un formulaire de type POST. Voici un exemple implémentant une méthode alternative à la (classique) variable `$_POST`.

```
<form method="post">
<input type="text" name="a" value="" />
<br />
<input type="radio" name="b" value="0" id="i_b0" />
<label for="i_b0">0</label>
<input type="radio" name="b" value="1" id="i_b1" />
<label for="i_b1">1</label>
<br />
<input type="checkbox" name="c" value="coche" />
<br />
<input type="submit" name="d" value="envoyer" />
</form>
<?php
$request = file_get_contents("php://input");
echo $request;
?>
```

Le résultat est une chaîne de caractères du type

a=test&b=1&c=coche&d=envoyer

Les filtres

Le gestionnaire `php://` offre des possibilités particulièrement intéressantes qui permettent d'effectuer des traitements élémentaires aux données lues, au travers d'un flux, au fur et à mesure de leur disponibilité. Il s'agit en quelque sorte de filtrer les données.

Ce peut être le seul recours pour une fonction comme `readfile()` et peux permettre d'éviter un traitement à posteriori avec `file_get_contents()` ou `file()`

La liste des filtres disponibles est donnée par la fonction `stream_get_filters()`.

stream_get_filters()

Retourne la liste des filtres disponibles sous la forme d'un tableau indexé.

Syntaxe : `array stream_get_filters(void)`

`retour` Tableau indexé contenant la liste des filtres disponibles.

Utilisez simplement ce script pour voir quels sont les filtres que vous pouvez utiliser.

```
<?php
print_r(stream_get_filters());
?>
```

Vous devriez observer un résultat de ce type :

```
Array
(
   [0] => convert.iconv.*
   [1] => string.rot13
   [2] => string.toupper
   [3] => string.tolower
   [4] => string.strip_tags
   [5] => convert.*
   [6] => zlib.*
)
```

Tableau 9.5 : Les filtres utilisables par défaut avec le gestionnaire php://filter

Filtres	Descriptions
convert.iconv.*	Change l'encodage de la chaîne de caractères.
string.rot13	Effectue un encodage ROT13 à la chaîne de caractères. L'encodage ROT13 décale les caractères de 13 dans l'alphabet. Exemple : a devient n, b devient o, c devient p.
string.toupper	Retourne la chaîne en majuscule.
string.tolower	Retourne la chaîne en minuscule.

Filtres	Descriptions
string.strip_tags	Filtre les tags de formatage HTML.
zlib.*	Utilise les fonctions de la zlib pour compresser ou décompresser les données à la volée.
convert.base64-*	convert.base64-encode permet d'encoder en base64. convert.base64-decode permet de décoder une chaîne de caractères encodée en base64.
convert.quoted-printable-*	convert.quoted-printable-encode permet de convertir une chaîne dans le format 'quoted-printable'. Le format 'quoted-printable' est souvent utilisé dans le corps des mail ou dans le posts de newsgroup. convert.quoted-printable-decode permet de décoder une chaîne au format 'quoted-printable'.

Pour utiliser ces filtres vous devez suivre la syntaxe suivante:

php://filter/

Applique un filtre en lecture ou écriture du fichier

Syntaxe :	`php://filter/[read=$filtreLecture\|write=$filtreEcriture] /resource=$ressource`
`$filtreLecture`	Paramètre optionnel indiquant le filtre à appliquer lors de la lecture.
`$filtreEcriture`	Paramètre optionnel indiquant le filtre à appliquer lors de l'écriture.
`$ressource`	Ressource du flux à filtrer, cela peut-être un simple nom de fichier ou une ressource s'appuyant sur les protocoles `http://`, `ftp://`, `compress.zlib://`, etc..

Exemple d'utilisation du gestionnaire `php` avec le paramètre `filter`.

```
<?php
// Recupère la page d'accueil de php.net
// Renvoie le contenu dans la chaîne $contenu en appliquant
// un filtre qui retourne tous les caractères en majuscule.
$contenu =
✂ file_get_contents("php://filter/read=string.toupper/resource=http://php.net");

// Création du fichier compressé php.net.gz avec le contenu
// récupéré précédemment
$fp = fopen("compress.zlib://php.net.gz","w")
      or die("Impossible de créer le fichier.");
fwrite($fp, $contenu);
fclose($fp);

// Affichage du fichier décompressé avec application d'un filtre
// retournant tous les caracteres en majuscules
```

```
readfile("php://filter/read=string.tolower/resource=compress.zlib://php.net.gz");
?>
```

Il est possible d'associer un filtre à un flux de façon à ce que le filtre en question soit utilisé pour chaque opération effectuée sur ce flux. On peut utiliser pour cela les fonctions `stream_filter_append()` ou `stream_filter_prepend()`. Dans le premier cas, le filtre est rattaché à la suite des filtres déjà attribués au flux. La seconde fonction permet de spécifier un filtre qui sera placé en tête de liste des filtres à utiliser. Les filtres en haut de la liste étant utilisés en premier.

stream_filter_append()

Attribue un filtre à un flux passé en paramètre. Il est placé en queue de liste des filtres déjà attachés au flux.

Syntaxe :	`bool stream_filter_append(ressource $flux, string $filtre [, int $comportement[, mixed $parametre]])`
`$flux`	Ressource vers la flux retournée par `fopen()` ou `fsockopen()`.
`$filtre`	Filtre à utiliser.
`$comportement`	Dans le cas ou le flux a été ouvert en mode lecture et écriture, ce paramètre permet de fixer si le filtre doit s'appliquer en lecture, en écriture ou dans les deux cas. `STREAM_FILTER_READ` : Appliquer le filtre uniquement en lecture. `STREAM_FILTER_WRITE` : Appliquer le filtre uniquement en écriture. `STREAM_FILTER_ALL` : Appliquer le filtre dans tous les cas.
`$parametre`	Paramètres optionnels pouvant être nécessaires à l'utilisation du filtre `$filtre`.
`retour`	Retourne `TRUE` si le filtre a bien été rattaché au flux et `FALSE` dans le cas contraire.

stream_filter_prepend()

Attribue un filtre à un flux passé en paramètre. Il est placé en tête de la liste des filtres déjà attachés au flux.

Syntaxe :	`bool stream_filter_prepend(ressource $flux, string $filtre [, int $comportement [, mixed $parametre]])`
`$flux`	Ressource vers la flux retournée par `fopen()` ou `fsockopen()`.
`$filtre`	Filtre à utiliser.
`$comportement`	Dans le cas ou le flux a été ouvert en mode lecture et écriture, ce paramètre permet de fixer si le filtre doit s'appliquer en lecture, en écriture ou dans les deux cas.

	`STREAM_FILTER_READ` : Appliquer le filtre uniquement en lecture. `STREAM_FILTER_WRITE` : Appliquer le filtre uniquement en écriture. `STREAM_FILTER_ALL` : Appliquer le filtre dans tous les cas.
`$parametre`	Paramètres optionnels pouvant être nécessaires à l'utilisation du filtre `$filtre`.
`retour`	Retourne `TRUE` si le filtre a bien été rattaché au flux et `FALSE` dans le cas contraire.

```
<?php
// Ouverture d'un fichier en lecture et écriture
$fp = fopen("bible.txt", "w+");

// On attache le filtre ROT13 uniquement en écriture
stream_filter_append($fp, "string.rot13", STREAM_FILTER_WRITE);

// On attache au flux, le filtre string.toupper
stream_filter_append($fp, "string.toupper", STREAM_FILTER_WRITE);
// On attache au flux, le filtre string.tolower en fin de liste
// Par conséquent le filtre string.toupper n'aura aucun effet
stream_filter_append($fp, "string.tolower", STREAM_FILTER_WRITE);

// Ecriture de données dans le fichier
fwrite($fp, "Damien HEUTE\n");
fwrite($fp, "Thomas HEUTE\n");
fwrite($fp, "Laurent GUEDON\n");
fwrite($fp, "Pierre-Emmanuel MULLER\n");

rewind($fp);

// Lecture du fichier
while ($ligne=fread($fp, 1024) ) {
    echo ($ligne);
}
fclose($fp);
?>
```

On obtient le résultat suivant :

```
qnzvra urhgr
gubznf urhgr
ynherag thrqba
cvreer-rzznahry zhyyre
```

Si on modifie le code en spécifiant le filtre `string.tolower` en début de liste comme ceci :

```
<?php
// Ouverture d'un fichier en lecture et écriture
$fp = fopen("bible.txt", "w+");

// On attache le filtre ROT13 uniquement en écriture
stream_filter_append($fp, "string.rot13", STREAM_FILTER_WRITE);
```

```
// On attache au flux, le filtre string.toupper
stream_filter_append($fp, "string.toupper", STREAM_FILTER_WRITE);
// On attache au flux, le filtre string.tolower en début de liste
// Par conséquent ce filtre n'aura aucun effet (écrasé par string.toupper)
stream_filter_prepend($fp, "string.tolower", STREAM_FILTER_WRITE);

// Ecriture de données dans le fichier
fwrite($fp, "Damien HEUTE\n");
fwrite($fp, "Thomas HEUTE\n");
fwrite($fp, "Laurent GUEDON\n");
fwrite($fp, "Pierre-Emmanuel MULLER\n");

rewind($fp);

// Lecture du fichier
while ($ligne=fread($fp, 1024) ) {
    echo ($ligne);
}
fclose($fp);
?>
```

On obtient alors :

QNZVRA URHGR
GUBZNF URHGR
YNHERAG THRQBA
CVREER-RZZNAHRY ZHYYRE

Modifions encore notre scripte en modifiant le comportement du `filtre string.rot13` en `STREAM_FILTER_ALL`.

Cette fois la lecture du scripte utilise le filtre ROT13, ainsi le résultat de son exécution est le suivant :

DAMIEN HEUTE
THOMAS HEUTE
LAURENT GUEDON
PIERRE-EMMANUEL MULLER

Contexte de ressource

Vous pouvez spécifier un contexte lors de l'appel à une ressource. Par exemple vous pouvez passer des paramètres à une URL. Pour cela, il est possible d'enregistrer votre contexte à l'aide de la fonction `stream_context_create()`.

stream_context_create()

Enregistre un contexte de flux utilisé ensuite lors de l'appel aux fonctions `file()`, `file_get_contents()` ou `fopen()`.

Syntaxe :	`ressource stream_context_create([array $contexte])`
`$contexte`	Tableau optionnel fournissant les contextes. De la forme `$tableau[gestionnaire][option] = valeur`.
`retour`	Retourne une ressource a passer en paramètre des fonctions `file()`, `file_get_contents()` ou `fopen()`.

Exemple illustrant l'utilisation de `stream_context_create()` :

```
<?php

$postdata = "auteur1=Laurent%20GUEDON&".
        "auteur2=Damien%20HEUTE&".
        "auteur3=Pierre%20Emmanuel%20MULLER&".
        "auteur4=Thomas%20HEUTE\n";

$header = "User-Agent: Bible Browser V0.3\n".
        "Referer:http://www.monsite.com\n".
        "Content-Length:".strlen($postdata)."\n".
        "Content-Type: application/x-www-form-urlencoded\n\n".
        $postdata;

$contexte = array(
    'http'=>array (
        'method' => 'POST',
        'header' => $header
    )
);

$fc = stream_context_create($contexte);

$fp = fopen('http://localhost/bible/recup.php', 'r', false, $fc);

fpassthru($fp);
fclose($fp);
?>
```

Ce script appelle une page `recup.php` décrite ci-dessous :

```
<?php
    echo "Les variables postées : \n";
    print_r($_POST);
    echo "\n----\n";
    echo "Provenance:".$_SERVER["HTTP_REFERER"]."\n";
    echo "Type de contenu:".$_SERVER["CONTENT_TYPE"];
?>
```

L'exécution du script donne le résultat suivant :

```
Les variables postées :
Array
(
   [auteur1] => Laurent GUEDON
   [auteur2] => Damien HEUTE
   [auteur3] => Pierre Emmanuel MULLER
   [auteur4] => Thomas HEUTE
)

----
Provenance:http://www.monsite.com
Type de contenu:application/x-www-form-urlencoded
```

On peut imaginer facilement les multiples usages qui en découlent.

Pour ne pas avoir à spécifier la ressource de contexte dans les fonctions `file()`, `file_get_contents()` ou `fopen()` vous pouvez utiliser la fonction `stream_context_get_default()`.

stream_context_get_default()

Spécifie les options de contexte par défaut lors des appels aux fonctions `file()`, `file_get_contents()` ou `fopen()`.

Syntaxe :	`ressource stream_context_get_default([array $contexte])`
`$contexte`	Tableau optionnel fournissant les contextes. De la forme `$tableau[gestionnaire][option] = valeur`.
`retour`	Retourne une ressource pouvant être passée en paramètre des fonctions `file()`, `file_get_contents()` ou `fopen()`.

```
<?php
$postdata = "auteur1=Laurent%20GUEDON&".
        "auteur2=Damien%20HEUTE&".
        "auteur3=Pierre%20Emmanuel%20MULLER&".
        "auteur4=Thomas%20HEUTE\n";

$header = "User-Agent: Bible Browser V0.3\n".
        "Referer:http://www.monsite.com\n".
        "Content-Length:".strlen($postdata)."\n".
        "Content-Type: application/x-www-form-urlencoded\n\n".
        $postdata;


$contexte = array(
    'http'=>array (
        'method' => 'POST',
        'header' => $header
```

```
    )
);

$fc = stream_context_get_default($contexte);

$fp = fopen('http://localhost/bible/recup.php', 'r');

fpassthru($fp);
fclose($fp);
?>
```

Le résultat est identique à l'exemple précédent :

```
Les variables postées :
Array
(
   [auteur1] => Laurent GUEDON
   [auteur2] => Damien HEUTE
   [auteur3] => Pierre Emmanuel MULLER
   [auteur4] => Thomas HEUTE
)

----
Provenance:http://www.monsite.com
Type de contenu:application/x-www-form-urlencoded
```

Autre solution, utiliser la fonction `stream_context_set_option()`.Cette fonction permet de fixer une option de contexte après l'appel à la fonction `stream_context_create()`.

stream_context_set_option()

Fixe une option de contexte à la ressource de flux passée en paramètre `file()`, `file_get_contents()` ou `fopen()`.

Syntaxe :	`bool stream_context_set_option(ressource $contexte, string $gestionnaire, string $option, mixed $valeur)`
`$contexte`	Ressource du contexte créée à l'aide de la fonction `stream_context_create()`.
`$gestionnaire`	Identifiant du gestionnaire affecté.
`$option`	Option du gestionnaire modifiée
`$valeur`	Nouvelle valeur de l'option
`retour`	Retourne `TRUE` si l'opération a été effectuée avec succès et `FALSE` dans le cas contraire.

```
<?php
$postdata = "auteur1=Laurent%20GUEDON&".
        "auteur2=Damien%20HEUTE&".
```

```
        "auteur3=Pierre%20Emmanuel%20MULLER&".
        "auteur4=Thomas%20HEUTE\n";

$header = "User-Agent: Bible Browser V0.3\n".
        "Referer:http://www.monsite.com\n".
        "Content-Length:".strlen($postdata)."\n".
        "Content-Type: application/x-www-form-urlencoded\n\n".
        $postdata;

$fc = stream_context_create();

stream_context_set_option($fc, 'http', 'method', 'POST');
stream_context_set_option($fc, 'http', 'header', $header);

$fp = fopen('http://localhost/bible/recup.php', 'r', false, $fc);

fpassthru($fp);
fclose($fp);
?>
```

On obtient la même chose que précédemment, c'est à dire :

```
Les variables postées :
Array
(
   [auteur1] => Laurent GUEDON
   [auteur2] => Damien HEUTE
   [auteur3] => Pierre Emmanuel MULLER
   [auteur4] => Thomas HEUTE
)

----
Provenance:http://www.monsite.com
Type de contenu:application/x-www-form-urlencoded
```

A tout moment, nous pouvons récupérer les informations de contexte, simplement en utilisant la fonction `stream_context_get_options()`.

stream_context_get_options()

Retourne les options de contexte dans un tableau associatif.

Syntaxe :	`array stream_context_get_options(ressource $contexte)`
`$contexte`	Ressource du contexte créée à l'aide de la fonction `stream_context_create()`.
`retour`	Tableau associatif contenant les informations de contexte.

```
<?php
$postdata = "auteur1=Laurent%20GUEDON&".
        "auteur2=Damien%20HEUTE&".
        "auteur3=Pierre%20Emmanuel%20MULLER&".
        "auteur4=Thomas%20HEUTE\n";

$header = "User-Agent: Bible Browser V0.3\n".
        "Referer:http://www.monsite.com\n".
        "Content-Length:".strlen($postdata)."\n".
        "Content-Type: application/x-www-form-urlencoded\n\n".
        $postdata;

$fc = stream_context_create();

echo "Options de contexte \n";
print_r(stream_context_get_options($fc));

stream_context_set_option($fc, 'http', 'method', 'POST');
stream_context_set_option($fc, 'http', 'header', $header);
echo "\nOptions de contexte, une fois modifiées \n";
print_r(stream_context_get_options($fc));
?>
```

Voici le résultat de ce script :

```
Options de contexte
Array
(
)

Options de contexte, une fois modifiées
Array
(
    [http] => Array
        (
            [method] => POST
            [header] => User-Agent: Bible Browser V0.3
Referer:http://www.monsite.com
Content-Length:106
Content-Type: application/x-www-form-urlencoded

auteur1=Laurent%20GUEDON&auteur2=Damien%20HEUTE&auteur3=Pierre%20Emmanuel%20
✂ MULLER&auteur4=Thomas20HEUTE

        )

)
```

Travailler avec les flux

Copier d'une ressource à une autre

La fonction `stream_copy_to_stream()` permet de copier le contenu pointé par une ressource de flux vers une autre ressource. Même si les gestionnaires utilisés par les deux flux sont différents.

stream_copy_to_stream()

Copie les données d'un flux vers un autre.

Syntaxe :	`int stream_copy_to_stream(ressource $source, ressource $destination [, int $taille])`
`$source`	Ressource du flux source.
`$destination`	Ressource du flux destination
`$taille`	Taille en octet des données à copier. Par défaut toutes les données seront intégralement copiées.
`retour`	Retourne le nombre d'octet ayant été copié si l'opération a été effectuée avec succès et `FALSE` dans le cas contraire.

```
<?php
$src = fopen('http://www.php.net', 'r');
$dest = fopen('ftp://laurent:motdepasse@ftp.serveur.org/php.net.html', 'w');
$nboct = stream_copy_to_stream($src, $dest);
echo $nboct." octets ont été copiés sur le serveur\n";
?>
34775 octets ont été copiés sur le serveur
```

Récupérer le contenu d'un flux

Récupérer l'ensemble

On peut également avoir besoin de récupérer le contenu d'un flux dans une chaîne. La fonction `stream_get_contents()` permet de récupérer le reste d'un contenu d'un flux.

stream_get_contents()

Récupère le reste du contenu d'un flux dans une chaîne.

Syntaxe :	`string stream_get_contents(ressource $flux [, int $taille])`
`$flux`	Ressource du flux que l'on considère.

`$taille`	Taille en octet des données à récupérer. Par défaut toutes les données seront intégralement retournées.
`retour`	Chaîne de caractère.

L'exemple suivant retourne le fichier index du site Php.net comme le ferait la fonction `readfile()`.

```
<?php
$flux = fopen('http://www.php.net', 'r');
$content = stream_get_contents($flux);
echo $content;
?>
```

Récupérer une ligne

On peut également lire les lignes les unes à la suite des autres. On utilise pour cela la fonction `stream_get_line()`.

stream_get_line()

Récupère une ligne de donnée dans un flux.

Syntaxe :	`string stream_get_line (ressource $flux, int $taille, string $fin)`
`$flux`	Ressource du flux que l'on considère.
`$taille`	Taille en octet de la ligne à récupérer. La lecture se termine lorsque $taille octets ont été lus ou lorsque la fin de la ligne fixée par le paramètre $fin a été rencontrée.
`$fin`	Chaîne de caractère représentant la fin d'une ligne (exemple : \n).
`retour`	Chaîne de caractère.

L'exemple ci-dessous permet de récupérer uniquement l'entête HTML du document présent en index du site `PHP.NET`.

```
<?php
$flux = fopen('http://www.php.net', 'r');
$ligne =  stream_get_line($flux, 4048, "</head>");
echo $ligne;
?>
```

Informations sur une ressource

Une fois la ressource ouverte, il peut-être intéressant de récupérer des informations sur celle-ci. La fonction `stream_get_meta_data()` permet de retrouver bon nombre d'informations sur la ressource de flux.

stream_get_meta_data()

Retourne les informations sur une ressource de flux.

Syntaxe :	`array stream_get_meta_data(ressource $flux)`
`$flux`	Ressource ouverte avec les fonctions `fopen()`, `fsockopen()` ou `pfsockopen()`.
`retour`	Retourne un tableau associatif comportant les différentes information sur le flux.

```
<?php
$fp = fopen('ftp://tild:passe@ftp.monserveur.org/php.net.html', 'r');
print_r(stream_get_meta_data($fp));
fclose($fp);
?>
```

L'exemple précédent retourne un résultat du type :

```
Array
(
   [wrapper_data] =>
   [wrapper_type] => ftp
   [stream_type] => tcp_socket
   [mode] => r+
   [unread_bytes] => 0
   [seekable] =>
   [uri] => ftp://tild:passe@ftp.monserveur.org/php.net.html
   [timed_out] =>
   [blocked] => 1
   [eof] =>
)
```

Ajouter des gestionnaires

Il serait dommage de se limiter à ces quelques protocoles supportés. Le langage PHP, dans sa grande souplesse, permet de créer de nouveaux gestionnaires. Ainsi on peut imaginer que, désirant ouvrir un fichier dans un format que vous auriez convenu préalablement, vous puissiez indiquer votre gestionnaire aux fonctions de lecture et d'écriture de fichiers. De la sorte, il vous est plus commode d'effectuer les opérations élémentaires en utilisant simplement les fonctions classiques d'accès aux fichiers.

Nous allons implémenter ici un gestionnaire qui va nous permettre d'accéder à une base de données. Pour cela il faut faire appel à la fonction `stream_wrapper_register()` ou à son alias `stream_register_wrapper()`.

stream_wrapper_register()

Cette fonction permet d'enregistrer un nouveau protocole définit par une classe.

Syntaxe :	`bool stream_wrapper_register(string $protocole, string $classe)`
`$protocole`	Nom du protocole à définir.
`$classe`	Nom de la classe utilisée pour implémenter le protocole.
`retour`	Retourne `TRUE` si le protocole à bien été enregistré et `FALSE` dans le cas contraire (Par exemple si le protocole est déjà enregistré).

Avant d'utiliser cette fonction, vous devez avoir implémenté une classe. Celle-ci comprend un certain nombre de méthodes qu'il vous faut prendre en compte.

Exemple PHP 5
L'exemple qui va suivre prend en compte les modifications de PHP5 pour la partie objet. Cela a pour conséquence que des modifications doivent être apportées pour adapter la classe à une version antérieure (PHP 4.3).

Il faut dans un premier temps permettre l'ouverture du flux et indiquer dans la classe les opérations qui doivent être effectuées à la suite d'un appel à la fonction `fopen()`. Pour ce faire nous utiliserons la méthode `stream_open()`.

(Classe Stream Personnalisée)->stream_open()

Cette commande interprète l'appel à la fonction `fopen()` et analyse le chemin du flux qui doit être traité ainsi que le mode d'ouverture.

Syntaxe :	`bool stream_open(string $chemin, string $mode, int $options, string $cheminReel)`
`$chemin`	Chemin vers le flux à traiter. Il faut utiliser la fonction `parse_url()` pour analyser la chaîne de caractères.
`$mode`	Mode d'ouverture du flux voir la fonction `fopen()` pour la liste des modes disponibles. Au développeur d'analyser le mode pour gérer les traitements utilisables.
`$option`	Fournit des options particulières pour permettre l'ouverture de la ressource de flux : `STREAM_USE_PATH` : Indique que la ressource est à rechercher dans le chemin courant ou à l'aide de la configuration de l'`include_path` (voir le chapitre sur php.ini).

	STREAM_REPORT_ERRORS : Le développeur de la classe est responsable de la levé des erreurs qui pourrait survenir (Voir la fonction trigger_error()).
$cheminReel	Chemin réel de la ressource.
retour	Retournez TRUE si l'opération a été effectuée avec succès et FALSE dans le cas contraire.

Voici un exemple implémentant uniquement l'ouverture pour examiner les données passées à la fonction stream_open() lorsque l'on exécute fopen().

```
<?php
class dbStream {
    function stream_open($chemin, $mode, $options='', &$cheminReel='') {
        echo $chemin;
        echo "\n";
        echo $mode;
        return TRUE;
    }
}
stream_wrapper_register('ex1','dbStream')
    or die("Impossible d'enregistrer ce protocole !");

    $flux = fopen("ex1://monchemin", "r");
?>
```

Le résultat est le suivant :

```
ex1://monchemin
r
```

Pour implémenter la fermeture du flux, il faut mettre en place la méthode stream_close().

(Classe Stream Personnalisée)->stream_close()

Méthode appelée lors de l'exécution de la fonction fclose().

Syntaxe : `void stream_close(void)`

Nous allons maintenant mettre en place la connexion à notre base de données. Pour l'exemple nous allons créer une base de données à l'aide du script SQL suivant :

```
CREATE DATABASE 'bible' ;
CREATE TABLE 'auteurs' (
'id' INT NOT NULL AUTO_INCREMENT ,
'nom' VARCHAR( 255 ) NOT NULL ,
'prenom' VARCHAR( 255 ) NOT NULL ,
PRIMARY KEY ( 'id' )
) COMMENT = 'Table des auteurs de la Bible du PHP';
```

La connexion à notre table s 'effectuera en indiquant l'url suivante :

`mysql://dbUser:dbPasse@serveur/base`

`dbUser` étant un utilisateur autorisé à accéder à la base de données.

`dbPasse` étant le mode de passe de l'utilisateur.

`serveur` indique l'adresse du serveur de la base.

`base` est le nom de la base de données utilisée pour la connexion.

`table` enfin, est le nom de la table ou se trouve nos données.

Voici comment implémenter la classe correspondante :

```
<?php
class fluxMysql {
    private $db;
    private $rs;
    private $table;

    function stream_open($chemin, $mode, $options='', &$cheminReel='') {
        $url = parse_url($chemin);
        $this->db = mysql_connect($url['host'], $url['user'], $url['pass']);
        // On sépare la base et la table
        list($base, $this->table) = split("\.", $url['path']);
        // On supprime le "/" devant le nom de la base
        $base = substr($base, 1-strlen($base));
        mysql_select_db($base);

        // On exécute la requête de sélection de la table
        $this->rs = mysql_query('select * from '.$this->table, $this->db);

        return TRUE;
    }

    function stream_close() {
        return mysql_close($this->db);
    }
}

stream_wrapper_register('mysql','fluxMysql')
    or die("Impossible d'enregistrer ce protocole !");

// Ouverture de la connexion
$flux = fopen("mysql://root@localhost/bible.auteurs", "r");

// Terminer la connexion
fclose($flux);
?>
```

Bien sur cette fonction n'affiche rien pour le moment. Pour se faire vous devez mettre en place la méthode qui sera appelée lors de l'exécution d'une fonction `fread()` ou `fgets()`. Pour se faire, vous devez implémenter `stream_read()` dans votre classe.

(Classe Stream Personnalisée)->stream_read()

Méthode exécuté lors de l'appel de la fonction `fread()` ou `fgets()`.

Syntaxe :	`string stream_read(int $nombre)`
`$nombre`	Nombre d'octets qui doivent être lu (en fonction du pointeur courant).
`retour`	Chaîne de caractère correspondant à la position du pointeur courant et de taille correspondant au nombre, `$nombre`, de caractères lus. Si aucun caractère ne peut-être retourné, la fonction retourne `FALSE`.

Pour illustrer par un exemple la mise en place de cette méthode, nous allons ajouter deux enregistrements à notre base de données à l'aide du script SQL suivant :

```
INSERT INTO 'auteurs' ('nom' , 'prenom' )
    VALUES ('GUEDON', 'Laurent');
INSERT INTO 'auteurs' ('nom' , 'prenom' )
    VALUES ('HEUTE', 'Thomas');
```

Maintenant, implémentons notre méthode de lecture.

```
function stream_read($nombre) {
    $this->position++;
    $chaineretour = '';
    $row = mysql_fetch_array($this->rs);
    for ($i=0; $i<mysql_num_fields($this->rs); $i++) {
        $chaineretour .= $row[mysql_field_name($this->rs, $i)];

        // Format la chaîne retournée
        // Tabulation entre les champs
        // Retour à la ligne lorsque tous
        // les champs ont été lus
        if($i+1==mysql_num_fields($this->rs)){
            $chaineretour .= "\n";
        } else {
            $chaineretour .= "\t";
        }
    }
    return $chaineretour;
}
```

Si vous essayez d'effectuer une lecture à ce niveau, vous verrez le message suivant apparaître :

```
Warning: fread() [function.fread]: fluxMysql::stream_eof is not implemented!
```

En effet, vous devez implémenter une méthode `stream_eof()` avant d'effectuer n'importe quelle lecture sur votre flux.

(Classe Stream Personnalisée)->stream_eof()

Méthode indiquant si le pointeur du flux se trouve à la fin du fichier ou non. Utilisé dans le cas de l'exécution de la fonction `feof()`.

Syntaxe :	`bool stream_eof(void)`
`retour`	Retourne `TRUE` si le pointeur se trouve en fin de lecture de la ressource.

Pour notre exemple, nous implémenterons cette méthode comme ceci :

```
function stream_eof() {
    // Vérifie si toutes les entrées ont été lues
    if($this->position < $this->num_rows) {
        // Si non, retourne FALSE
        return FALSE;
    } else {
        // Dans le cas contraire, renvoie TRUE
        return TRUE;
    }
}
```

Sans oublier d'ajouter ceci au début de la classe

```
private $position = 0;
```

La lecture peut maintenant s'effectuer de la façon suivante :

```
stream_wrapper_register('mysql','fluxMysql')
    or die("Impossible d'enregistrer ce protocole !");

// Ouverture de la connexion
$flux = fopen("mysql://root@localhost/bible.auteurs", "r");
while($c = fread($flux, 2048)) {
    echo $c;
}
// Terminer la connexion
fclose($flux);
```

On obtient le résultat suivant :

```
1....GUEDON....Laurent
2....HEUTE.....Thomas
```

Un exemple sans octet
Vous remarquerez qu'ici, nous n'avons pas traité le second paramètre de la fonction `fread()`. Mais il est simple de découper le résultat pour ne retourner qu'une partie de la chaîne en fonction du nombre d'octets désiré.

Vous pouvez remarquer que la lecture s'effectue simplement, le pointeur se déplaçant d'un enregistrement à l'autre à chaque appel de la fonction `fread()`.


ATTENTION

PHP 4.3 oui, PHP 5 non...
Cet exemple bien que fonctionnel dans les versions 4.3 et supérieur ne fonctionne pas dans la version récente de PHP5. Le bug a été reporté et cela devrait être rétabli dans les prochaines versions de PHP 5.

Vous savez qu'il est simple de connaître la position du pointeur à chaque moment. Vous utilisez `ftell()` en indiquant la ressource utilisée et la fonction vous retourne la position actuelle du pointeur. Pour implémenter un équivalent pour votre gestionnaire, vous devez mettre en place une méthode nommée `stream_tell()` dans votre classe.

(Classe Stream Personnalisée)->stream_tell()

Méthode appelée lors de l'exécution de la fonction `ftell()`. Vous devez retourner la position du pointeur.

Syntaxe :	`int stream_tell(void)`
`retour`	Vous devez renvoyer la position du pointeur de type entier.

Voici comment dans notre exemple, la méthode `stream_tell()` retourne la position actuelle du pointeur.

```
function stream_tell() {
    return $this->position;
}
```

Puis ajouter ceci :

```
rewind($flux);
echo ftell($flux)."\n";
```


REMARQUE

Bug de jeunesse
Cette fonctionnalité n'est pas utilisable dans PHP pour le moment, en effet, à l'heure de la rédaction de cette ouvrage (version 5.0.2), `ftell()` n'utilise pas le valeur retournée par de la méthode `stream_tell()`. Peut-être pour la version de PHP 5.1...

Lorsque vous utilisez les fonctions de lecture vous pouvez facilement déplacer le pointeur à l'aide de la fonction `fseek()`. Là encore vous pouvez implémenter une méthode qui va vous permettre l'usage de cette fonction. Il faut mettre en place la méthode `stream_seek()`.

(Classe Stream Personnalisée)->stream_seek()

Permet de modifier la position du pointeur du flux. Cette méthode est utilisée lors de l'appel à la fonction `fseek()`. Vous devez implémenter la méthode `stream_tell()` auparavant.

Syntaxe :	`bool stream_seek(int $offset, int $origine)`
`$offset`	Nouvelle position du pointeur depuis l'origine `$origine`.
`$origine`	Point de départ pour calculer la nouvelle position. Voir les paramètre utilisés par la fonction `fseek()`.
`retour`	La méthode doit retourner `TRUE` si le déplacement a bien été effectué et `FALSE` dans le cas contraire.

Placez cette méthode dans votre classe :

```
function stream_seek($offset, $origine=SEEK_SET) {
    switch($origine) {
        // L'origine du déplacement est 0
        case SEEK_SET:
            $dep = 0;
            break;
        // L'origine du déplacement est la position
        // courante
        case SEEK_CUR:
            $dep = $this->position;
            break;
        // L'origine du déplacement est la fin du
        // flux
        case SEEK_END:
            $dep = mysql_num_rows($this->rs);
           break;
    }

    // Déplacement
    $this->position = $offset+$dep;
    return mysql_data_seek($this->rs, $this->position);
}
```

Ainsi que l'appel à la fonction `fseek()` dans votre programme :

```
echo "Déplacer le pointeur sur la seconde entrée :";
fseek($flux, 1);
echo fread($flux, 2048)."\n";
```

Maintenant observons de quelle manière nous pouvons utiliser `fwrite()` pour écrire des données dans la table. Pour cela nous devons mettre en place la méthode `stream_write()`.

(Classe Stream Personnalisée)->stream_write()

Cette méthode est appelée en réponse à l'exécution de la fonction `fwrite()`. Vous devez implémenter la méthode en fonction des données passées en argument.

Syntaxe :	`int stream_write(string $donnee)`
`$donnee`	Chaîne de caractère qu'il faut traiter dans la fonction.
`retour`	Vous devez retourner la longueur, en octet, des données qui ont été prise en compte par votre fonction.

Voici pour notre exemple, comment nous pouvons ajouter un enregistrement. Il faut dans un premier temps implémenter la méthode :

```
function stream_write($donnee) {
    $data = explode ("\n", $donnee);
    $nboct = 0;
    // Formate la chaîne d'insertion des données
    // Celles-ci sont passées à la fonction fwrite()
    // sous la forme :
    // champ1=donnée 1\nchamp2=donnée 2\netc...
    $formatstr1 = $formatstr2 = "(";
    for ($i=0; $i<count($data); $i++) {
        list($champ, $valeur) = explode ("=", $data[$i]);
        $formatstr1 .= $champ;
        $formatstr2 .= "'".$valeur."'";
        $nboct = $nboct + strlen($valeur);
        // Ajoute une virgule le cas échéant
        if(count($data) > $i+1) {
            $formatstr1 .= ", ";
            $formatstr2 .= ", ";
        }
    }
    $formatstr1 .= ")";
    $formatstr2 .= ")";

    if(mysql_query("INSERT INTO ".$this->table." ".$formatstr1.
                   " VALUES ".$formatstr2, $this->db)) {
        // On execute de nouveau la requete afin de récupérer le dernier
        // enregistrement
        mysql_free_result($this->rs);
        $this->rs = mysql_query('select * from '.$this->table, $this->db);

        // On place le pointeur sur la dernière entrée
        $this->stream_seek(0, SEEK_END);
        return $nboct;
    } else {
        return FALSE;
    }
}
```

Et ajoutons ce code à la fin de notre script:

```
echo "Enregistrement d'une nouvelle entrée : ";
echo fwrite($flux, "nom=HEUTE\nprenom=Damien");
echo " caractères ont été enregistré\n";
// On écrit les entrées pour vérifier que l'écriture
// à bien été effectuée
rewind($flux);
echo "Position actuelle du pointeur :".ftell($flux)."\n";
while($c = fread($flux, 2048)) {
    echo "\n";
    echo "entrée :";
    echo $c;
}
```

Le résultat est maintenant le suivant :

```
Position actuelle du pointeur :17
1   GUEDON  Laurent

Position actuelle du pointeur :32
2   HEUTE   Thomas
Position actuelle du pointeur :0
Déplacer le pointeur sur la seconde entrée :2   HEUTE   Thomas

Enregistrement d'une nouvelle entrée : 11 caractères ont été enregistré
Position actuelle du pointeur :0

entrée :1   GUEDON  Laurent

entrée :2   HEUTE   Thomas

entrée :3  HEUTE   Damien
```

Vous pouvez observer que nous ne prenons pas en compte le mode d'ouverture de la ressource. En effet, alors que nous avions sélectionné l'ouverture en lecture seule, il est possible d'écrire des données dans la table. En conséquence, c'est au développeur qu'il incombe la responsabilité de vérifier les accès à la ressource en fonction du mode d'ouverture de celle-ci. Nous pouvons spécifier ceci directement dans la méthode correspondante sans oublier d'assigner le mode à la propriété `$mode` de notre objet.

Nous déclarons dans un premier temps une nouvelle propriété :

```
private $mode;
```

Ensuite, il faut lui assigner une valeur lors de l'appel à la fonction `fopen()` :

```
function stream_open($chemin, $mode, $options='', &$cheminReel='') {
    $url = parse_url($chemin);
    $this->db = mysql_connect($url['host'], $url['user'], $url['pass']);
    // On sépare la base et la table
    list($base, $this->table) = split("\.", $url['path']);
    // On supprime le "/" devant le nom de la base
```

```
    $base = substr($base, 1-strlen($base));
    mysql_select_db($base);

    // On exécute la requête de sélection de la table
    $this->rs = mysql_query('select * from '.$this->table, $this->db);
    $this->mode = $mode;
    return TRUE;
}
```

Enfin exécuter le test correspondant dans la méthode `fwrite()` :

```
function stream_write($donnee) {
    if($this->mode == "r") return FALSE;
    $data = explode ("\n", $donnee);
    $nboct = 0;
    // Formate la chaîne d'insertion des données
    // Celles-ci sont passées à la fonction fwrite()
....
```

Si vous exécutez le script, vous observez qu'à présent, le mode d'écriture "`r`" empêche la création de nouvelles entrées dans la table. Vous pouvez ajouter un contrôle dans la méthode `fread()` pour également empêcher la lecture lorsque le mode d'ouverture de la ressource est en écriture seule ("Write Only").

La fonction `fstat()` peut, elle aussi, être implémentée. Même s'il est évident qu'elle ne peut pas forcement retourner toutes les données que l'on peut trouver en effectuant un `fstat()` sur un système de fichier UNIX. La procédure `stream_stat()` est utilisée à chaque appel de cette fonction.

(Classe Stream Personnalisée)->stream_stat()

Doit retourner le tableau associatif avec les clefs suivantes : `[dev]`, `[ino]`, `[mode]`, `[nlink]`, `[uid]`, `[gid]`, `[rdev]`, `[size]`, `[atime]`, `[mtime]`, `[ctime]`, `[blksize]`, `[blocks]`

Syntaxe :	`array stream_stat(void)`
`retour`	Retourne un tableau associatif comportant une ou plusieurs valeurs.

Dans notre exemple, nous allons mettre en oeuvre la procédure pour permettre à la fonction `fstat()` de renseigner la date de création, la date de dernière modification ainsi que la taille de la table (en comptant les indexes).

```
function rettimestamp($datStr) {
    if(ereg("([0-9]{4})-([0-9]{2})-([0-9]{2}) ".  // Dates
        "([0-9]{2}):([0-9]{2}):([0-9]{2})",       // Heures
        $datStr, $regmatch)) {
        return  mktime($regmatch[4], $regmatch[5], $regmatch[6],
                    $regmatch[2], $regmatch[3], $regmatch[1]);
    } else {
        return FALSE;
```

```
    }
}

function stream_stat() {
    $sqlinfo = mysql_query('SHOW TABLE STATUS');
    $rsTableInfo = mysql_fetch_array($sqlinfo);

    $tableinfo = array(
        "size" =>  $rsTableInfo["Index_length"] + $rsTableInfo["Data_length"],
        "mtime" => $this->rettimestamp($rsTableInfo["Update_time"]),
        "ctime" => $this->rettimestamp($rsTableInfo["Create_time"])
    );
    return $tableinfo;
}
```

Il ne nous reste plus qu'à appeler notre fonction `fstat()` et observer le résultat:

Listing 9.43 : stream_open.php

```
echo "Informations sur la table :\n";
$stat = fstat($flux);
echo "- Taille:".round($stat["size"]/1000, 1)." Ko\n";
echo "- Dernière modification:".date('j/m/Y G:i:s', $stat["mtime"])."\n";
echo "- Date de création:".date('j/m/Y G:i:s', $stat["ctime"])."\n";
```

```
Informations sur la table :
- Taille:2.7 Ko
- Dernière modification:24/09/2004 13:49:18
- Date de création:15/09/2004 17:05:50
```

La fonction `unlink()` est utilisée pour supprimer les liens vers une ressource. Depuis PHP 5.0, il est possible d'implémenter dans un gestionnaire une méthode pouvant gérer cette fonction. Il faut pour cela définir la méthode `unlink()`.

(Classe Stream Personnalisée)->unlink()

Décrire dans la méthode, la suppression de la ressource définie par son URL. Elle ne peut être implémentée qu'avec les versions 5 et supérieures du langage PHP.

Syntaxe :	`bool unlink(string $chemin)`
`$chemin`	URL de la ressource à supprimer
`retour`	Doit retourner `TRUE` si la ressource a bien été supprimée et `FALSE` dans le cas contraire.

Voici un exemple expliquant la suppression d'une ressource de type table à l'aide de la fonction `unlink()`.

Listing 9.44 : stream_unlink.php

```
<?php
class fluxMysql {
     function unlink($chemin) {
        $url = parse_url($chemin);
        $db = mysql_connect($url['host'], $url['user'], $url['pass']);
        // On sépare la base et la table
        list($base, $table) = split("\.", $url['path']);
        // On supprime le "/" devant le nom de la base
        $base = substr($base, 1-strlen($base));
        mysql_select_db($base);
        mysql_query('DROP TABLE '.$table, $db);
        mysql_close($db);
     }
}

stream_wrapper_register('mysql','fluxMysql')
    or die("Impossible d'enregistrer ce protocole !");

// Connexion
$db = mysql_connect("localhost", "root", "");
mysql_select_db("bible");
// Création d'une table
mysql_query('CREATE TABLE 'unlink' ('.
    ''ch1' VARCHAR( 10 ) NOT NULL ,'.
    ''ch2' VARCHAR( 10 ) NOT NULL ,'.
    ''ch3' VARCHAR( 10 ) NOT NULL)', $db);

// Insertion des données
mysql_query("INSERT INTO 'unlink' VALUES('test1', 'test1', 'test1')");
mysql_query("INSERT INTO 'unlink' VALUES('test2', 'test2', 'test2')");
mysql_query("INSERT INTO 'unlink' VALUES('test3', 'test3', 'test3')");

// Lecture des entrées dans la table
$rs = mysql_query("SELECT * FROM 'unlink'");
while ($row = mysql_fetch_row($rs)) {
    echo "Entrée : \n";
    echo "\t".$row[0]."\n";
    echo "\t".$row[1]."\n";
    echo "\t".$row[2]."\n\n";
}
mysql_close($db);

// Suppression de la table via la fonction unlink()
unlink("mysql://root@localhost/bible.unlink");

//Vérification de la suppression de la table
// Connexion
$db = mysql_connect("localhost", "root", "");
mysql_select_db("bible");
// Lecture des entrées dans la table
```

```
$rs = mysql_query("SELECT * FROM 'unlink'")
    or die ("Impossible de récupérer les entrées dans cette table");
while ($row = mysql_fetch_row($rs)) {
    echo "Entrée : ";
    echo "\t".$row[0]."\n";
    echo "\t".$row[1]."\n";
    echo "\t".$row[2]."\n\n";
}
mysql_close($db);
?>
```

Et voici le résultat :

```
Entrée :
    test1
    test1
    test1

Entrée :
    test2
    test2
    test2

Entrée :
    test3
    test3
    test3

Impossible de récupérer les entrées dans cette table
```

Tous comme les fichiers, vous pouvez être amené à renommer vos ressources. Dans le cas d'une base, vous pouvez modifier le nom de celle-ci. La méthode `rename()` est appelée à chaque exécution de la fonction du même nom (Si vous spécifiez, bien sur, votre gestionnaire dans l'URL).

(Classe Stream Personnalisée)->rename()

Renomme ou modifie le chemin vers votre ressource. Méthode à implémenter uniquement avec les versions 5 et supérieures du langage PHP.

Syntaxe :	`bool rename(string $ancienneRess, string $nouvelleRess)`
`$ancienneRess`	Chemin vers l'ancienne ressource disponible
`$nouvelleRess`	Nouvelle ressource
`retour`	Retournez `TRUE` si l'opération a été exécutée avec succès et `FALSE` dans le cas contraire.

Voici, comme à l'habitude, un exemple d'utilisation :

Listing 9.45 : stream_rename.php

```
<?php
class fluxMysql {
     function rename($anChemin, $nvChemin) {
        $url1 = parse_url($anChemin);
        $url2 = parse_url($nvChemin);
        // On ne prends pas en compte ici l'url complète
        // du nouveau chemin et on suppose que la même base de données
        // est utilisée pour la modification
        $db = mysql_connect($url1['host'], $url1['user'], $url1['pass']);
        // On sépare la base et la table
        list($base1, $table1) = split("\.", $url1['path']);
        list($base2, $table2) = split("\.", $url2['path']);
        // On supprime le "/" devant le nom de la base
        $base1 = substr($base1, 1-strlen($base1));
        mysql_select_db($base1);

        mysql_query('ALTER TABLE '.$table1.' RENAME TO '.$table2, $db);
        mysql_close($db);
     }
}

stream_wrapper_register('mysql','fluxMysql')
    or die("Impossible d'enregistrer ce protocole !");

// Connexion
$db = mysql_connect("localhost", "root", "");
mysql_select_db("bible");
// Création d'une table
mysql_query('CREATE TABLE 'nomnaze' ('.
    ''ch1' VARCHAR( 10 ) NOT NULL ,'.
    ''ch2' VARCHAR( 10 ) NOT NULL ,'.
    ''ch3' VARCHAR( 10 ) NOT NULL)', $db);

// Insertion des données
mysql_query("INSERT INTO 'nomnaze' VALUES('test1', 'test1', 'test1')");
mysql_query("INSERT INTO 'nomnaze' VALUES('test2', 'test2', 'test2')");
mysql_query("INSERT INTO 'nomnaze' VALUES('test3', 'test3', 'test3')");

// Lecture des entrées dans la table
$rs = mysql_query("SELECT * FROM 'nomnaze'");
echo "Il y a ".mysql_num_rows($rs)." Entrée(s) dans la table naze !\n";
mysql_close($db);

// Renommons la table
rename("mysql://root@localhost/bible.nomnaze",
    "mysql://root@localhost/bible.nomquitue"
    );
```

```
//Vérification que la table porte bien le nouveau nom
// Connexion en utilisant l'ancien nom
$db = mysql_connect("localhost", "root", "");
mysql_select_db("bible");
// Lecture des entrées dans la table en utilisant l'ancien nom
if($rs = mysql_query("SELECT * FROM 'nomnaze'")) {
    echo "L'ancien nom est utilisé !";
} elseif($rs = mysql_query("SELECT * FROM 'nomquitue'")) {
    echo "Nous utilisons maintenant le nouveau nom de la table !\n";
    while ($row = mysql_fetch_row($rs)) {
        echo "Entrée : \n";
        echo "\t".$row[0]."\n";
        echo "\t".$row[1]."\n";
        echo "\t".$row[2]."\n\n";
    }
}
mysql_close($db);
?>
```

```
Il y a 3 Entrée(s) dans la table naze !
Nous utilisons maintenant le nouveau nom de la table !
Entrée :
    test1
    test1
    test1

Entrée :
    test2
    test2
    test2

Entrée :
    test3
    test3
    test3
```

Comme les pour fonctions précédentes, la fonction `mkdir()` qui est utilisée pour créer un répertoire, peut-être ici utilisé pour effectuer une toute autre opération. Nous devons pour cela implémenter une nouvelle méthode dans notre classe. Celle-ci est tous simplement la méthode `mkdir()`.

(Classe Stream Personnalisée)->mkdir()

Méthode à implémenter pour exécuter les commandes à réaliser à la suite de l'appel à la fonction `mkdir()`. Elle ne peut être implémentée qu'avec les versions 5 et supérieures du langage PHP.

Syntaxe : `function mkdir(string $chemin, int $mode, int $options)`

`$chemin`	URL possédant le filtre à appliquer ainsi que les informations permettant de traiter la ressource.
`$mode`	Le mode est utilisé pour indiquer les droits sur le dossier à créer.
`$options`	Peut-être indiquées les options suivantes : `STREAM_REPORT_ERRORS` : Indique si les erreurs doivent être levées par le gestionnaire. `STREAM_MKDIR_RECURSIVE` : Indique s'il faut permettre la récursivité de la fonction `mkdir()`.
`retour`	Indiquez `TRUE` si les commandes ont été réalisées avec succès et `FALSE` dans le cas contraire.

Voici un exemple d'implémentation de la méthode `mkdir()`.

Listing 9.46 : stream_mkdir.php

```
<?php
class fluxMysql {
    function mkdir($chemin, $mode=0777, $options=0) {
        $url = parse_url($chemin);
        $db = mysql_connect($url['host'], $url['user'], $url['pass']);
        $base = substr($url['path'], 1-strlen($url['path']));

        $retour = mysql_query('CREATE DATABASE '.$base, $db);

        mysql_close($db);

        return $retour;
    }
}

stream_wrapper_register('mysql','fluxMysql')
    or die("Impossible d'enregistrer ce protocole !");
if (mkdir("mysql://root@localhost/dbrmdir")) {
    echo "La base a été créée avec succès !";
} else {
    echo "Impossible de créer la base de données !";
}
?>
```

```
La base a été créée avec succès !
```

Vous pouvez vérifier que la base de données est bien en place à l'aide de `phpMyAdmin` ou du client `MySQL`.

Si vous essayez de créer une nouvelle fois la base de données, le script vous retournera :

```
Impossible de créer la base de données !
```

Cette base une fois créée, vous pouvez la supprimer en SQL ou implémenter la méthode `rmdir()` dans votre classe pour qu'elle effectue l'opération.

(Classe Stream Personnalisée)->rmdir()

Méthode utilisée à l'exécution de la fonction `rmdir()`. Elle ne peut être implémentée qu'avec les versions 5 et supérieures du langage PHP.

Syntaxe :	`bool rmdir(string $chemin, int $options)`
`$chemin`	URL possédant le filtre à appliquer ainsi que les informations permettant de supprimer la ressource.
`$options`	Indiquez `STREAM_REPORT_ERRORS` si vous désirez relever les erreurs à l'aide du gestionnaire.
`retour`	Indiquez `TRUE` si les commandes de suppression ont été réalisées avec succès et `FALSE` dans le cas contraire.

Voici le code précédent arrangé pour permettre la suppression de la base que vous venez de créer.

```
<?php
class fluxMysql {
    function mkdir($chemin, $mode=0777, $options=0) {
        $url = parse_url($chemin);
        $db = mysql_connect($url['host'], $url['user'], $url['pass']);
        $base = substr($url['path'], 1-strlen($url['path']));

        $retour = mysql_query('CREATE DATABASE '.$base, $db);

        mysql_close($db);

        return $retour;
    }
    function rmdir($chemin, $option) {
        $url = parse_url($chemin);
        $db = mysql_connect($url['host'], $url['user'], $url['pass']);
        $base = substr($url['path'], 1-strlen($url['path']));

        $retour = mysql_query('DROP DATABASE '.$base, $db);

        mysql_close($db);
        return $retour;
    }
}

stream_wrapper_register('mysql','fluxMysql')
    or die("Impossible d'enregistrer ce protocole !");

    // Création de la base de données
if (mkdir("mysql://root@localhost/dbrmdir")) {
    echo "La base a été créée avec succès !\n";
} else {
    echo "Impossible de créer la base de données !\n";
}
```

```
    // Suppression de la base de données
if (rmdir("mysql://root@localhost/dbrmdir")) {
    echo "La base a été supprimée avec succès !\n";
} else {
    echo "Impossible de supprimer la base de données !\n";
}
?>
La base a été créée avec succès !
La base a été supprimée avec succès !
```

Voyons maintenant comment nous pourrions lister les tables contenues dans une base de données. Pour cela nous allons implémenter dans notre gestionnaire la méthode `dir_opendir()`.

(Classe Stream Personnalisée)->dir_opendir()

Méthode appelée lors de l'utilisation de la fonction `opendir()`.

Syntaxe :	`bool dir_opendir(string $chemin, int $options)`
`$chemin`	URL indiquant le chemin vers la ressource à traiter.
`$options`	Indiquez `STREAM_REPORT_ERRORS` si vous désirez relever les erreurs à l'aide du gestionnaire.
`retour`	Indiquez `TRUE` si la ressource a été ouverte avec succès et `FALSE` dans le cas contraire.

Nous avons également besoin de créer une méthode `dir_readdir()` qui sera appelé lors de chaque exécution de la fonction `readdir()`.

(Classe Stream Personnalisée)->dir_readdir()

Méthode retournant le nom du prochain "fichier" ouvert. Celle-ci est appelé à l'exécution de la fonction `readdir()`. Cette méthode s'utilise conjointement avec `dir_opendir()`.

Syntaxe :	`string dir_readdir(void)`
`retour`	Retourne une chaîne représentant le nom du prochain fichier.

N'oublions pas qu'il est nécessaire (parfois) de libérer la ressource utilisée. La méthode `dir_closedir()` est là pour ça !

(Classe Stream Personnalisée)->dir_closedir()

Méthode appelée lors de l'exécution de la fonction `closedir()`. Vous l'implémenterez de façon à libérer la ressource ouverte par `dir_opendir()`.

Syntaxe :	`bool dir_closedir(void)`
`retour`	Indiquez `TRUE` en cas de succès et `FALSE` dans le cas contraire.

Implémentation de ces méthodes pour récupérer les tables contenues dans une base de données.

Listing 9.47 : dir_opendir.php

```
<?php
class fluxMysql {
    private $base;
    var $table;

    function dir_opendir($chemin, $option=0) {
        $url = parse_url($chemin);
        $this->base = mysql_connect($url['host'], $url['user'], $url['pass']);

        $base = substr($url['path'], 1-strlen($url['path']));
        if($this->table = mysql_list_tables($base, $this->base)) {
            return TRUE;
        } else {
            return FALSE;
        }
    }

    function dir_readdir() {
        if ($row = mysql_fetch_row($this->table)) {
            return $row[0];
        } else {
            return FALSE;
        }
    }

    function dir_closedir() {
        return mysql_close($this->base);
    }
}

stream_wrapper_register('mysql','fluxMysql')
    or die("Impossible d'enregistrer ce protocole !");

// Création d'une base de données pour l'exemple
// Connexion au serveur
$db = mysql_connect("localhost", "root", "");
mysql_query('CREATE DATABASE dir_opendir', $db);
```

```
// Connexion à cette base de données
mysql_select_db("dir_opendir");

// Création de 10 tables
for ($i=0; $i<10; $i++) {
    mysql_query('CREATE TABLE ’ex'.$i.'’ ('.
        '’ch1’ VARCHAR( 10 ) NOT NULL ,'.
        '’ch2’ VARCHAR( 10 ) NOT NULL ,'.
        '’ch3’ VARCHAR( 10 ) NOT NULL)', $db);
}
mysql_close($db);

// Liste les tables de la base de données
$flux = opendir("mysql://root@localhost/dir_opendir")
    or die("Impossible d'ouvir la ressource !");

echo "Liste des tables : \n";

while($table = readdir($flux)) {
    echo "-> ".$table."\n";
}

closedir($flux);

// Suppression de la base
$db = mysql_connect("localhost", "root", "");
mysql_query('DROP DATABASE dir_opendir', $db);
mysql_close($db);
?>
```

Le résultat de l'exécution de ce script :

```
Liste des tables :
-> ex0
-> ex1
-> ex2
-> ex3
-> ex4
-> ex5
-> ex6
-> ex7
-> ex8
-> ex9
```

Enfin, il est possible de réinitialiser le pointeur du flux si vous implémenter la méthode `dir_rewinddir()`.

(Classe Stream Personnalisée)->dir_rewinddir()

Méthode appelée lors de l'exécution de la fonction `rewinddir()`. A vous d'écrire les commandes permettant de réinitialiser le pointeur du flux à sa position principale.

Syntaxe :	`bool dir_rewinddir(void)`
`retour`	Retournez la valeur `TRUE` en cas de succès et `FALSE` dans le cas contraire.

Voici la source finale de cet exemple :

```
<?php
class fluxMysql {
    private $base;
    var $table;

    function dir_opendir($chemin, $option=0) {
        $url = parse_url($chemin);
        $this->base = mysql_connect($url['host'], $url['user'], $url['pass']);

        $base = substr($url['path'], 1-strlen($url['path']));
        if($this->table = mysql_list_tables($base, $this->base)) {
            return TRUE;
        } else {
            return FALSE;
        }
    }

    function dir_readdir() {
        if ($row = mysql_fetch_row($this->table)) {
            return $row[0];
        } else {
            return FALSE;
        }
    }

    function dir_closedir() {
        return mysql_close($this->base);
    }

    function dir_rewinddir() {
        return mysql_data_seek($this->table, 0);
    }
}

stream_wrapper_register('mysql','fluxMysql')
    or die("Impossible d'enregistrer ce protocole !");


// Création d'une base de données pour l'exemple
// Connexion au serveur
```

```
$db = mysql_connect("localhost", "root", "");
mysql_query('CREATE DATABASE dir_opendir', $db);

// Connexion à cette base de données
mysql_select_db("dir_opendir");

// Création de 10 tables
for ($i=0; $i<10; $i++) {
    mysql_query('CREATE TABLE ’ex'.$i.'’ ('.
        '’ch1’ VARCHAR( 10 ) NOT NULL ,'.
        '’ch2’ VARCHAR( 10 ) NOT NULL ,'.
        '’ch3’ VARCHAR( 10 ) NOT NULL)', $db);
}
mysql_close($db);

// Liste les tables de la base de données
$flux = opendir("mysql://root@localhost/dir_opendir")
    or die("Impossible d'ouvir la ressource !");

echo "Liste des tables : \n";
while($table = readdir($flux)) {
    echo "-> ".$table."\n";
}

// On reinitialise le pointeur à la position 0
rewinddir($flux);

// Et on recommence
echo "\nListe des tables (encore !): \n";
while($table = readdir($flux)) {
    echo "-> ".$table."\n";
}

closedir($flux);

// Suppression de la base
$db = mysql_connect("localhost", "root", "");
mysql_query('DROP DATABASE dir_opendir', $db);
mysql_close($db);
?>
```

Avec le résultat suivant :

```
Liste des tables :
-> ex0
-> ex1
-> ex2
-> ex3
-> ex4
-> ex5
-> ex6
-> ex7
```

```
-> ex8
-> ex9

Liste des tables (encore !):
-> ex0
-> ex1
-> ex2
-> ex3
-> ex4
-> ex5
-> ex6
-> ex7
-> ex8
->.ex9
```

Chapitre 10

L'utilisation des bases de données

Si vous souhaitez stocker des informations et pouvoir, par la suite, les trier, les compter, les filtrer, alors vous serez certainement amené à utiliser une base de données.

Les bases de données peuvent être utilisées pour gérer un répertoire téléphonique ; dans ce cas, vous y stockerez probablement des noms, prénoms, adresses et numéros de téléphone et, la plupart du temps, vous filtrerez ces informations par nom. Elles peuvent également être utilisées pour stocker les messages d'un forum, qui pourront alors être triés par date ou par fil de discussion.

Comme vous le pressentez, il y a de nombreux domaines d'applications ayant recours aux bases de données.

Le choix d'une base de données n'est pas simple, il peut vous être dicté par votre hébergeur, par votre supérieur, par le coût, par les performances, par votre "religion" informatique... PHP 5 simplifie grandement l'utilisation de petites bases de données en intégrant SQLite qui ne nécessite aucune installation ni création d'utilisateurs.

Avant de voir comment utiliser une base de données (avec PHP), détaillons ce qu'est une base de données (nous ne nous intéresserons ici qu'aux bases de données relationnelles).

10.1. Introduction aux bases de données

Une base de données peut être assimilée à un ensemble de fichiers (dans lesquels sont stockées les informations). Ces derniers sont gérés uniquement par un logiciel serveur (il n'est absolument pas question que vous les manipuliez directement). Tout comme vos scripts PHP sont mis à disposition du public par un serveur Internet, vos données sont mises à disposition par un serveur de bases de données. Les utilisateurs, quant à eux, devront utiliser un logiciel client pour manipuler ces données.

Les différents éditeurs de bases de données fournissent ainsi systématiquement le client et le serveur de bases de données. En règle générale, le client d'une base de données ne sera capable de communiquer qu'avec le serveur pour lequel il a été conçu. Comme nous le verrons par la suite, il existe toutefois des standards permettant de communiquer avec plusieurs types de serveurs de bases de données.

Comme nous l'avons dit, l'utilisateur devra accéder aux données en utilisant le logiciel client approprié. Or, si le logiciel diffère d'un serveur de bases de données à l'autre, il existe des familles de bases de données qui utilisent plus ou moins le même langage, à savoir le langage SQL.


REMARQUE

Interface graphique
Il existe également des clients ayant une interface graphique utilisable directement via votre navigateur. Pour MySQL, vous pourrez par exemple utiliser phpMyAdmin.


RENVOI

Reportez-vous à l'annexe "phpMyAdmin" pour plus de détails.

10.2. Introduction au langage SQL

Le langage SQL (Structured Query Language = langage de requêtes structurées) est censé être normalisé (notamment avec les normes SQL92 et SQL99), mais la plupart des éditeurs l'adaptent à leur sauce.

Ce langage s'appuie sur une représentation des données sous forme de tables. Une table peut être assimilée à un tableau. Chaque table est composée de champs (les colonnes du tableau) et d'enregistrements (les lignes du tableau). Dans le cas d'un répertoire téléphonique, chaque enregistrement représentera un individu, et il y aura un champ pour le nom, un champ pour le prénom, etc.

Une base de données peut (et c'est quasiment toujours le cas) contenir plusieurs tables liées les unes aux autres.

Il est à noter qu'un même serveur de bases de données peut héberger plusieurs bases de données (autrement dit plusieurs ensembles de tables).

Le typage

Une des forces des bases de données SQL sur les fichiers est, entre autres, que les champs sont typés. Il n'est donc pas question de tenter de mettre une chaîne de caractères là où un entier est attendu, ce qui assure une certaine cohérence dans les informations. En revanche, il faudra évidemment bien réfléchir au choix du type de données (ce qui ne constitue généralement pas une grosse difficulté).

Cela peut avoir son importance, notamment lors d'opérations de tri. Ainsi, des nombres stockés dans un champ de type texte ne seront pas triés de la même manière que s'ils sont dans un champ de type entier. En effet, la chaîne de caractères "10" est alphanumériquement plus petite (avant) que "2" (le premier caractère de "10" étant avant le premier caractère de "2"), alors que le nombre 2 est plus petit que le nombre 10.

Le choix du type est également important en terme d'occupation du disque : un nombre stocké sous forme d'une chaîne de caractères prend nécessairement plus de place que s'il est stocké sous sa forme binaire. Inutile, de même, de choisir un type sur 3 octets pour un nombre entier compris entre 0 et 255 (tenant donc sur un unique octet).

Les contraintes

Le langage SQL permet également d'imposer des contraintes (autres que le type) sur les données, ceci toujours afin d'assurer la cohérence des informations. Il est, par exemple, ainsi possible d'obliger à ce qu'un champ soit renseigné, ou bien à ce que la valeur donnée appartienne à une liste prédéfinie (ce dernier point n'est toutefois pas vrai pour tous les serveurs de bases de données).

10.3. Les relations entre tables

Les règles de conception d'une base de données peuvent un peu dérouter les néophytes, mais ce n'est finalement pas aussi compliqué que cela paraît (même s'il existe des méthodes bien savantes, la plus célèbre concernant les bases de données étant Merise). En effet, il n'est

généralement pas question d'avoir une base de données ne possédant qu'une seule table, dans laquelle on mettrait toutes les informations (comme dans un fourre-tout). Non. Généralement, une base est constituée de plusieurs tables reliées entre elles par un ou plusieurs champs identifiant de façon unique un enregistrement de la table.

Explication. Imaginez que vous souhaitez constituer une base de données de films. Dans ce cas, vous serez amené à indiquer le nom du film, le nom du réalisateur ainsi que la liste des acteurs. Mais, vous comprenez bien qu'il n'est pas question d'avoir une table (comme la suivante) avec, pour chaque enregistrement, un triplet (film, réalisateur, acteur), sachant qu'il y a plusieurs acteurs par film et que nous serions obligés de répéter le nom du réalisateur.

Tableau 10.1 : Exemple de table d'une base de données mal conçue

Film	Réalisateur	Acteur
Forrest Gump	Robert Zemeckis	Tom Hanks
Forrest Gump	Robert Zemeckis	Robin Wright Penn
Il faut sauver le soldat Ryan	Steven Spielberg	Tom Hanks
Il faut sauver le soldat Ryan	Steven Spielberg	Edward Burns

Nous pourrions éventuellement envisager de créer une table avec plusieurs champs *acteur* (*acteur1*, *acteur2*, etc.). Bien que ce soit généralement le premier réflexe des néophytes, cela est une très mauvaise idée. Ne serait ce que parce que l'on ne sait pas, a priori, combien il y a d'acteurs (au maximum) par film.

Non, comme nous l'avons déjà dit, il faut alors faire plusieurs tables. Dans notre cas, une première approche (pas encore satisfaisante) peut consister à avoir une table de couples film/réalisateur et une table de couples film/acteur. Mais, dans ce cas, nous serions à nouveau obligé de dupliquer les noms des films et des acteurs (si ces noms sont saisis manuellement, les risques de fautes de frappes sont importants).

La solution consiste à créer une table *film*, une table *réalisateur* et une table *acteur*, ainsi que des tables de liaisons. Les valeurs de champs dupliquées seront alors remplacées par des références à un champ d'une table.

Ce qui donne, dans notre cas :

Tableau 10.2 : Table Film

FilmId	Film
1	Forrest Gump
2	Il Faut Sauver le Soldat Ryan

Tableau 10.3 : Table Acteur

ActeurId	Acteur
1	Tom Hanks
2	Robin Wright Penn
3	Edward Burns

Tableau 10.4 : Table de liaison Film-Acteur

FilmId	ActeurId
1	1
1	2
2	1
2	3

Tableau 10.5 : Table Réalisateur

RealisateurId	Realisateur
1	Robert Zemeckis
2	Steven Spielberg
3	Nora Ephron

Tableau 10.6 : Table de liaison Film-Réalisateur

FilmId	RealisateurId
1	1
2	2
3	3

À première vue, cela est devenu plutôt illisible et difficile à manipuler. Détrompez-vous ; les requêtes SQL vous permettront d'accéder simplement aux données. La difficulté que peut présenter la saisie des données dans ces tables sera généralement masquée par une interface graphique. Mais, surtout, nous avons grandement gagné en liberté. Il n'y a désormais plus de contraintes: Il est possible d'indiquer autant de noms d'acteurs et de réalisateurs que voulus par film. Aucun nom de film, d'acteur ou de réalisateur n'est dupliqué (il sera donc facile de les corriger si besoin).

Et enfin, détail qui a toute son importance, une base de données ainsi structurée occupera beaucoup moins de place que les premières versions envisagées. En effet, si l'on considère une base de 65 535 films (ce qui est relativement peu pour une base de données) faisant intervenir au plus un total de 65 535 acteurs et réalisateurs, alors les identifiants de film, acteur et réalisateur pourront être stockés sur 2 octets. Si l'on compte un réalisateur et cinq acteurs précisés par film, et si l'on suppose que les noms de film, d'acteur et de réalisateur nécessitent en moyenne 25 caractères, alors :

- La taille de la table *film* est 65 535*(2+25) = 1 769 445 octets.
- La taille de la table *acteur* est 65 535*(2+25) = 1 769 445 octets
- La taille de la table de liaison *film-acteur* est 65 535*5*(2+2) = 1 310 700 octets.
- La taille de la table réalisateur est 65 535*(2+25) = 1 769 445 octets.

- La taille de la table de liaison film-réalisateur est 65 535*(2+2) = 262 140 octets.

Soit un total de 6 881 175 octets = 6,56 Mo.

Alors qu'avec la première version proposée, la base de données aurait eu une taille de 65 535*5*(25+25+25) = 24 575 625 octets = 23,4 Mo.

Clés primaires et compteurs

L'identifiant unique d'un enregistrement dans une table est appelé la *clé primaire*. C'est assez souvent un champ numérique (comme évoqué dans l'exemple précédent) portant comme nom "`Id`" ou se terminant par "`Id`" (ou encore "`PK`" comme "Primary Key"). La clé primaire peut toutefois être un champ d'un autre type, ou encore être un ensemble de champs.

Dans le cas d'une clé primaire numérique du type de celle présentée dans l'exemple précédent, il est fort intéressant de pouvoir déterminer quelle est la valeur suivante disponible. Pour cela, un certain nombre de serveurs de bases de données proposent un type particulier (ex. : `SERIAL` pour PostgreSQL) ou "modificateur" de type (ex. : `AUTO_INCREMENT` pour MySQL) que l'on peut appeler "compteur" ou "type auto-incrémenté". Certains serveurs (ex. : Oracle) ne proposent pas ce type de facilité, mais il est alors possible d'obtenir le même effet avec une `SEQUENCE` et un `TRIGGER`.

Index

Pour les champs sur lesquels de nombreuses recherches sont effectuées, il est généralement recommandé d'associer un index. Cette opération est bien souvent implicitement appliquée sur les clés primaires, mais vous pourrez être amené à vous poser la question de l'intérêt à l'utiliser pour un autre de vos champs. Nous ne nous étendrons pas sur ce sujet, mais sachez que ceci permet d'accélérer grandement les recherches (même si cela entraîne un délai supplémentaire lors de l'ajout de données, et réclame quelques ressources disque et mémoire supplémentaires).

10.4. Le langage SQL

Il n'est pas dans notre intention de décrire, ici, l'ensemble des commandes SQL (il faudrait y consacrer un livre complet). Mais voici, tout de même, les principales ; celles qui, probablement, répondront à 90 % de vos besoins.

Si vous souhaitez tester immédiatement ces commandes, vous devez au préalable consulter la documentation de votre base de données afin d'identifier et de lancer le logiciel client. Une fois que vous avez accès au client de votre base de données, vous pouvez saisir ces commandes, en n'omettant pas (généralement) de les faire suivre d'un point-virgule (indiquant ainsi au client, non pas que vous souhaitez aller à la ligne, mais que vous avez fini de saisir votre requête).

Il est à noter que certaines de ces opérations nécessitent des droits (privilèges) particuliers ; vous ne serez donc pas nécessairement autorisé à les réaliser.

Création/suppression d'une base de données

La création d'une base de données n'est pas toujours réalisable depuis le logiciel client de la base de données. Pour certaines bases, cette opération doit être réalisée à partir d'un exécutable fourni avec le serveur.

CREATE DATABASE

Crée une base de données.

Syntaxe	`CREATE DATABASE nombase`
`nombase`	Nom de la base de données à créer.

La suppression de la base de données (attention aux fausses manipulations) se fait via la commande :

DROP DATABASE

Supprime une base de données.

Syntaxe	`DROP DATABASE nombase`
`nombase`	Nom de la base de données à supprimer.

Les types

Avant de passer à la suite, il est nécessaire de vous présenter les différents types supportés par les bases de données. Là encore, cela peut varier d'un serveur à l'autre, mais cela reste, dans les grandes lignes, sensiblement identique (nous nous sommes basés principalement sur les types MySQL). Certains types peuvent avoir des noms différents selon les serveurs. De ce fait, de nombreux serveurs acceptent plusieurs noms pour désigner un même type.

Les types numériques

Tableau 10.7 : Les types numériques entiers signés

Type	Valeur min.	Valeur max.	Taille en octets
`TINYINT`	-128	127	1
`SMALLINT` `INT2`	-32768	32767	2
`MEDIUMINT`	-8388608	8388607	3

Type	Valeur min.	Valeur max.	Taille en octets
INT INT4 INTEGER	-2147483648	2147483647	4
BIGINT INT8	-9223372036854775808	9223372036854775807	8

Il est également possible d'utiliser des types entiers non signés en complétant le nom du type par UNSIGNED. Cela permet d'atteindre une valeur maximale plus grande (lorsque les noms négatifs ne sont pas utilisés).

Tableau 10.8 : Les types numériques entiers non signés

Type	Valeur min.	Valeur max.	Taille en octets
TINYINT UNSIGNED	0	255	1
SMALLINT UNSIGNED INT2 UNSIGNED	0	65535	2
MEDIUMINT UNSIGNED	0	16777215	3
INT UNSIGNED INT4 UNSIGNED INTEGER UNSIGNED	0	2147483647	4
BIGINT UNSIGNED INT8 UNSIGNED	0	18446744073709551615	8

Il est à noter que rares sont les bases de données qui proposent un type booléen.

Tableau 10.9 : Les types numériques décimaux

Type	Étendue	Taille en octets
FLOAT FLOAT4	[-3.402823466E+38, -1.175494351E-38] U {0} U [1.175494351E-38, 3.402823466E+38]	4
DOUBLE DOUBLE PRECISION REAL FLOAT8	[-1.7976931348623157E+308, -2.2250738585072014E-308] U {0} U [2.2250738585072014E-308, 1.7976931348623157E+308]	8
DECIMAL		
NUMERIC		

Les types texte

Tableau 10.10 : Les types texte

Type	Taille en octet	Description
`CHAR`	1	
`CHARACTER(X)` `CHAR(X)`	Dépend de X	Chaîne de taille fixe de X caractères, ne pouvant dépasser 255 caractères.
`CHARACTER VARYING(X)` `VARCHAR(X)`	Variable	Chaîne de taille variable ne pouvant dépasser X caractères (X ne peut dépasser 255).
`TINYBLOB` `TINYTEXT`	Variable	Chaîne de caractères limitée à 255 caractères.
`BLOB` `TEXT (MySQL)`	Variable	Chaîne de caractères limitée à 65 535 caractères.
`MEDIUMBLOB` `MEDIUMTEXT`	Variable	Chaîne de caractères limitée à 16 777 215 caractères.
`LONGBLOB` `LONGTEXT`	Variable	Chaîne de caractères limitée à 4 294 967 295 caractères.
`TEXT (PostgreSQL)`	Variable	Chaîne de caractères sans limite.

Les types date

Tableau 10.11 : Les types dates

Type	Format	Précision	Taille en octets
`DATE`	AAAA-MM-JJ	jour	
`DATETIME`	AAAA-MM-JJ hh:mm:ss	seconde	
`TIMESTAMP (MySQL)`	AAAAMMJJHHMMSS	seconde	
`TIMESTAMP`		microseconde	
`TIME (MySQL)`	HH:MM:SS	seconde	
`TIME (PostgreSQL)`		microseconde	4
`TIME WITH TIME ZONE (PostgresSQL)`		microseconde	4
`YEAR (MySQL)`		année	4

Création/suppression d'une table

Il existe de nombreuses variantes, d'une base de données à l'autre, au niveau des options (qui ne seront donc généralement pas précisées ici), mais les requêtes élémentaires restent les mêmes.

CREATE TABLE

Crée une table.

Syntaxe	`CREATE TABLE nomtable (champs)`
`nomtable`	Nom de la table à créer.
`champs`	Définition des champs de la table. (Voir ci-après)

Les champs de la table sont définis par une succession de séquences `"nomchamp type [attributs de champ]"` séparées par des virgules éventuellement suivies par les attributs de la table.

Un exemple tout simple de création de table permettant de stocker un identifiant de film, un nom et une date de sortie donne :

```
CREATE TABLE film (filmid INT2 UNSIGNED, film VARCHAR(64), datesortie DATE);
```

Les attributs des champs peuvent être :

`NOT NULL`	Pour indiquer que le champ ne peut pas être non renseigné (`NULL`).
`NULL`	Pour préciser que le champ peut être laissé non renseigné (`NULL`), (attribut par défaut).
`DEFAULT valeurpardefaut`	Pour préciser une valeur par défaut si le champ n'est pas renseigné (ou mis à `NULL`).
`PRIMARY KEY`	Pour indiquer qu'il s'agit d'une clé primaire.
`AUTO_INCREMENT`	(Uniquement valable pour MySQL). Pour préciser que, par défaut, ce champ doit prendre la valeur du dernier indice affecté + 1. Ce champ doit être une clé.

Si l'on reprend notre exemple, il est évident que le nom du film ne doit pas être laissé vide et que `filmid` est une clé primaire.

```
CREATE TABLE film (filmid INT2 UNSIGNED PRIMARY KEY,
                   film VARCHAR(64) NOT NULL,
                   datesortie DATE);
```

Comme nous l'avons également dit, il est souhaitable que le champ `filmid` soit auto-incrémenté ; la requête devrait être, sous MySQL :

```
CREATE TABLE film (filmid INT2 UNSIGNED PRIMARY KEY AUTO_INCREMENT,
                   film VARCHAR(64) NOT NULL,
                   datesortie DATE);
```

Les attributs de la table peuvent être :

`PRIMARY KEY (champ1, champ2, ...)` Pour préciser une clé primaire sur plusieurs champs.

`INDEX [nomindex] (champ1, champ2, ...)` Pour préciser une clé sur plusieurs champs. Un index sera alors créé, avec la possibilité de préciser son nom avec `nomindex`.

`UNIQUE [INDEX] [nomindex] (champ1, champ2, ...)` Pour contraindre l'unicité de l'ensemble (*champ1*, *champ2*, ..) dans la table.

Certaines bases utilisent le mot-clé `KEY` au lieu de `INDEX` (avec la même syntaxe).

Poursuivons notre exemple. Il n'est pas question de préciser plusieurs fois qu'un acteur donné a joué dans un film donné. Donc, pour éviter les doublons, la définition de la table de liaison *film-acteur* pourra être :

```
CREATE TABLE film2acteur (filmid INT2 NOT NULL,
                          acteurid INT2 NOT NULL,
                          UNIQUE(filmid, acteurid));
```

Vous vous êtes complètement trompé ? Vous ne voulez plus de cette table ? Pas de problème ! vous pouvez la supprimer avec l'instruction :

DROP TABLE

Suppression d'une table.

Syntaxe	`DROP TABLE nomtable`
`nomtable`	Nom de la table à supprimer (il est possible d'en préciser plusieurs séparés par une virgule).

Si, en revanche, vous souhaitez apporter une correction à une table existante, vous pouvez faire appel à :

ALTER TABLE

Modifie la structure d'une table existante.

Syntaxe	`ALTER nomtable modification`
`nomtable`	Nom de la table à modifier.
`modification`	Modification à apporter à la table (voir ci-après). (Il est possible d'apporter plusieurs modifications en les séparant par une virgule).

Les modifications qui peuvent être apportées sont :

- `ADD [COLUMN] champ` : pour ajouter un champ (en utilisant la même syntaxe que pour `CREATE TABLE`). Le mot-clé `COLUMN` n'a pas d'impact et est généralement optionnel. Il assure simplement la compatibilité avec les syntaxes d'autres serveurs de bases de données.
- `ADD PRIMARY KEY (champ1, champ2, ...)` : pour préciser une clé primaire (cf. `CREATE TABLE`).
- `ADD INDEX [nomindex] (champ1, champ2, ...)` : pour ajouter un index (cf. `CREATE TABLE`).
- `ADD UNIQUE [nomindex] (champ1, champ2, ...)` : pour ajouter une contrainte d'unicité (cf. `CREATE TABLE`).
- `DROP [COLUMN] nomchamp` : pour supprimer le champ "*nomchamp*". Le mot clé `COLUMN` n'a pas d'impact et est généralement optionnel. Il assure simplement la compatibilité avec les syntaxes d'autres serveurs de bases de données.
- `DROP PRIMARY KEY` : pour supprimer une définition de clé primaire.
- `DROP INDEX nomindex` : pour supprimer l'index "`nomindex`".
- `RENAME [AS|TO] nomtable2` : pour renommer la table en "*nomtable2*".

Ainsi, si nous voulons également préciser la durée du film (en minutes), et mettre, par défaut, 0, il est possible de reprendre la table créée précédemment pour faire :

```
ALTER TABLE film ADD duree INT2 DEFAULT 0
```

Ajouter des données

Nous allons maintenant pouvoir entrer dans le vif du sujet. Ajouter des données, c'est très simple. Pour cela, vous disposez de la commande :

INSERT

Ajoute un enregistrement dans une table.

Syntaxe	`INSERT INTO nomtable [(champ1, champ2, ...)] VALUES (valeur1, valeur2, ...)`
`nomtable`	Nom de la table dans laquelle insérer les données du nouvel enregistrement.
`champ1, champ2,`	Noms des champs que vous souhaitez préciser (les autres champs prendront leur valeur par défaut). Si vous ne spécifiez pas de liste de noms de champs, alors vous devrez préciser toutes les valeurs et dans l'ordre des champs.
`valeur1, valeur2,`	Valeurs des champs dans le même ordre que précisé par les noms des champs. Les chaînes de caractères et dates doivent être spécifiées entre apostrophes. Si la valeur contient une apostrophe, alors vous pouvez au

choix (et selon le serveur de bases de données) mettre deux apostrophes ou faire précéder l'apostrophe d'un anti-slash.

Une variante consiste à écrire INSERT INTO nomtable SET champ1=valeur1, champ2=valeur2, ..

Avec MySQL le mot-clé `INTO` est optionnel (mais ce n'est pas une raison pour l'omettre).

Si l'on reprend une table créée par

```
CREATE TABLE film (filmid INT2 UNSIGNED PRIMARY KEY,
                   film VARCHAR(64) NOT NULL)
```

il sera alors possible d'ajouter un enregistrement, soit avec la requête suivante :

```
INSERT INTO film VALUES (1, 'Forrest Gump');
```

soit en précisant les noms de tous les champs :

```
INSERT INTO film (filmid, film) VALUES (1, 'Forrest Gump');
INSERT INTO film (film, filmid) VALUES ('Forrest Gump', 1);
```

soit, notamment pour profiter du champ auto-incrémenté, avec :

```
INSERT INTO film VALUES (NULL, 'Forrest Gump');
INSERT INTO film (film) VALUES ('Forrest Gump');
```

Si le nom du film contient une apostrophe, alors la requête aura l'allure suivante :

```
INSERT INTO film (film) VALUES ('La vie n''est pas un long fleuve tranquille');
INSERT INTO film (film) VALUES ('La vie n\'est pas un long fleuve tranquille');
```

Mettre à jour des données

Il est bien évidemment possible de modifier des données ; pour cela, vous disposez de la commande `UPDATE`.

UPDATE

Met à jour un ensemble d'enregistrements.

Syntaxe	`UPDATE nomtable SET champ1=valeur1[, champ2=valeur2] [WHERE condition]`
`nomtable`	Nom de la table contenant les enregistrements à modifier.
`champX`	Nom du champ dont on veut modifier la valeur.
`valeurX`	Nouvelle valeur pour le champ.
`condition`	Critères de sélection des enregistrements devant être modifiés.

Si vous souhaitez apporter une modification sur l'ensemble de la table, vous pouvez lancer une requête du type :

```
UPDATE film SET film='Forrest Gump';
```

Dans ce cas, tous les films de la base porteront le même nom (ce n'est certainement pas ce que l'on veut, mais la requête aurait été similaire si nous avions voulu réinitialiser un champ servant, par exemple, à compter le nombre de clics sur un lien).

Il existe de nombreux opérateurs et fonctions pouvant être utilisés dans les expressions de conditions ou même encore qui peuvent être appliquées aux valeurs retournées. Même si nous vous en présenterons quelques-uns au fil de ce chapitre, il vous est conseillé de consulter la documentation de votre serveur de bases de données pour en avoir la liste exhaustive.

La condition la plus rudimentaire est, bien évidemment, l'égalité. Ainsi, si vous avez fait une faute d'orthographe dans le nom du film ayant l'identifiant 1, vous pourrez exécuter la requête :

```
UPDATE film SET film='Forrest Gump' WHERE filmid=1;
```

Mais il est également possible, par exemple, de modifier les noms de tous les titres de films commençant par "F" (même si, dans notre cas, cela n'a pas de sens), en utilisant l'instruction `LIKE` et le joker `%` (équivalent du `*` dans la plupart des systèmes d'exploitation, qui remplace un nombre quelconque de caractères).

```
UPDATE film SET film='Forrest Gump' WHERE film LIKE 'F%';
```

Le joker pour un caractère unique (équivalent du `?` dans la plupart des systèmes d'exploitation) est l'underscore '_'. Notez toutefois qu'il est possible de compléter la requête pour utiliser d'autres caractères comme joker.

Supprimer des données

Si vous souhaitez supprimer une donnée, faites appel à l'instruction `DELETE`.

DELETE

Supprime un ensemble d'enregistrements.

Syntaxe	`DELETE FROM nomtable [WHERE condition]`
`nomtable`	Nom de la table contenant les enregistrements à supprimer.
`condition`	Critères de sélection des enregistrements devant être modifiés.

Pour vider totalement la table *film*, vous n'aurez donc qu'à saisir la requête suivante :

```
DELETE FROM film;
```

Si vous souhaitez supprimer les enregistrements pour lesquels le nom du film n'est pas renseigné (cela ne peut pas arriver si vous avec créé la table avec le champ *film* défini comme `NOT NULL`), vous devrez utiliser :

```
DELETE FROM film WHERE film IS NULL;
```


REMARQUE

NULL est différent de la chaîne vide
Attention, il ne faut pas confondre un champ non renseigné (représenté par `NULL`) avec une chaîne de caractères vide (représentée par deux apostrophes consécutives).

Lire des données

Maintenant que nous avons vu comment ajouter des enregistrements, nous allons voir comment y accéder en lecture. Pour cela nous disposons de l'instruction `SELECT`, dont voici la syntaxe abrégée :

SELECT

Retourne un ensemble d'enregistrements.

Syntaxe	`SELECT [DISTINCT|ALL] champ1, champ2, ... FROM table1, table2,... [WHERE condition]`
`DISTINCT`	Pour ne retourner que des valeurs différentes (pas de doublons).
`ALL`	Pour autoriser les doublons (valeur par défaut).
`champ1, champ2,...`	Liste des champs à retourner.
`table1, table2`	Liste des tables impliquées dans la recherche.
`condition`	Condition que doivent remplir les enregistrements.

Pour avoir la liste de tous les noms des films connus de la base de données, vous n'aurez qu'à exécuter la requête :

```
SELECT film FROM film;
```

Si vous souhaitez également récupérer l'identifiant du film vous utiliserez alors plutôt :

```
SELECT filmid, film FROM film;
```

ce qui revient à afficher tous les champs des films, et qui peut se faire de façon plus générique avec :

```
SELECT * FROM film;
```

Si, par contre, vous ne souhaitez que le nom du film ayant pour identifiant la valeur 1, la requête devient :

```
SELECT film FROM film WHERE filmid=1;
```

Et, pour avoir la liste des films commençant par "F" :

```
SELECT film FROM film WHERE film LIKE 'F%';
```

Tout en continuant de nous intéresser à cette syntaxe de base, nous allons légèrement compliquer l'affaire afin de nous rapprocher d'une requête classique. Nous avons, en effet, vu qu'il était souvent préférable de stocker les informations dans de multiples tables liées entre elles par des clés primaires. Mais nous ne savons pas encore comment mettre en oeuvre ces liens. Pour cela, il suffit d'utiliser la requête `SELECT` sur plusieurs tables, et de préciser dans la clause `WHERE` la condition du lien.

Concrètement si l'on a deux tables, l'une stockant des noms de villes et l'autre des noms de magasins ainsi qu'un identifiant de ville (ville où se situe le magasin) créés avec les requêtes suivantes :

```
CREATE TABLE tableville (villeid INT4 PRIMARY KEY, ville VARCHAR(128));
CREATE TABLE tablemagasin (magasin VARCHAR(128), villeid INT4);
```

la requête permettant de retourner les couples (magasin, ville) sera alors

```
SELECT tablemagasin.magasin, tableville.ville FROM tablemagasin, tableville
       WHERE tablemagasin.villeid=tableville.villeid;
```

Vous noterez au passage qu'il est possible de préciser à quelle table appartient le champ précisé en utilisant la syntaxe "`nomtable.nomchamp`". Il n'est cependant pas nécessaire de préciser le nom de la table s'il n'y a pas de confusion possible, ce qui est vrai ici pour les champs "*magasin*" et "*ville*" mais pas pour le champ "*villeid*". La requête peut donc également s'écrire :

```
SELECT magasin, ville FROM tablemagasin, tableville
                     WHERE tablemagasin.villeid=tableville.villeid;
```

Si l'on reprend l'exemple des films dans lesquels interviennent trois tables, et si nous nous intéressons aux acteurs jouant dans ces films, cela donne :

```
SELECT film, acteur FROM film, acteur, film2acteur
                   WHERE film.filmid=film2acteur.filmid
                        AND film2acteur.acteurid;
```

Puisque vous semblez avoir compris, nous allons encore augmenter la difficulté. Il existe des cas où des tables sont liées plusieurs fois à une autre. C'est le cas notamment si vous souhaitez créer un dossier dans lequel sont stockés des noms d'individus, des lieux de naissance et des lieux de résidence. Là, la table contenant les noms de villes sera liée à la table des individus aussi bien par l'information *lieu de naissance* que par le *lieu de résidence*. Il faut alors faire intervenir deux fois le nom de la table *ville*, ce qui de prime abord donne "... `FROM individu, ville, ville` ...". Mais dans ce cas, il nous est impossible de distinguer les deux "instances" de la table *ville*. Pour pallier ce problème, il suffit de faire appel à des alias selon la syntaxe "`nomtable AS autrenom`". D'où la requête :

```
SELECT individu, villenaissance.ville, villeresidence.ville
    FROM individu, ville AS villenaissance, ville AS villeresidence
    WHERE individu.villenaissanceid=villenaissance.villeid
      AND individu.villeresidenceid=villeresidence.villeid;
```

Notez que les alias peuvent également être utilisés pour manipuler un nom de table plus court :

```
SELECT * FROM unnomdetabletroplong AS t, autretable WHERE t.id=autretable.id;
```

Maintenant que nous avons déjà bien progressé dans la connaissance de l'instruction `SELECT`, voici d'autres options permettant, entre autres, de trier les données.

SELECT

Retourne un ensemble d'enregistrements.

Syntaxe	`SELECT [DISTINCT|ALL] listechamp FROM listetable [WHERE condition] [GROUP BY listechampgb] [ORDER BY obchamp1 [DESC|ASC], obchamp2...]`
`listechamp`	Liste des champs à retourner.
`listetable`	Liste des tables impliquées dans la recherche.
`condition`	Condition que doivent remplir les enregistrements.
`GROUP BY`	Demande le regroupement des enregistrements.
`listechampgb`	Liste des champs selon lesquels les résultats de la requête doivent être groupés.
`ORDER BY`	Demande le tri des enregistrements.
`obchamp1`	Premier champ devant servir de champ de tri.
`DESC`	Effectue un tri décroissant.
`ASC`	Effectue un tri croissant (ordre de tri par défaut).

Pour obtenir un résultat trié selon un ou plusieurs champs, il faut donc faire appel à l'instruction `ORDER BY`.

Ainsi,

```
SELECT film FROM film ORDER BY film;
```

retourne la liste complète des films triés par ordre alphabétique ;

```
SELECT film FROM film ORDER BY film DESC;
```

retourne la liste complète dans l'ordre alphabétique inverse (de Z à A) ;

```
SELECT film, acteur FROM film, acteur, film2acteur
                    WHERE film.filmid=film2acteur.filmid AND film2acteur.acteurid
                    ORDER BY film, acteur;
```

retourne la liste complète dans l'ordre alphabétique des films et, pour chaque film, les acteurs sont classés par ordre alphabétique ;

```
SELECT film, acteur FROM film, acteur, film2acteur
                    WHERE film.filmid=film2acteur.filmid AND film2acteur.acteurid
                    ORDER BY film DESC, acteur;
```

retourne la liste complète dans l'ordre alphabétique inverse des films et, pour chaque film, les acteurs sont classés par ordre alphabétique.

Comme nous l'avons évoqué précédemment, il est possible d'appliquer des fonctions aux champs retournés, mais il est également possible de leur appliquer des fonctions d'agrégation. Il s'agit de fonctions s'opérant sur un ensemble d'enregistrements comme `COUNT()` pour compter le nombre d'enregistrements retournés, `SUM()` pour calculer la somme des valeurs d'un champ, et bien d'autres.

Ainsi, il est possible de déterminer le nombre de films que contient la base de données avec

```
SELECT COUNT(*) FROM film;
```

Mais il est également possible d'obtenir le nombre d'acteurs par film. Dans ce cas, il faut préciser que l'on souhaite faire le calcul par film avec l'instruction `GROUP BY` :

```
SELECT film, COUNT(acteur) FROM film, acteur, film2acteur
                    WHERE film.filmid=film2acteur.filmid AND film2acteur.acteurid
                    GROUP BY film;
```

Récupérer des informations sur une base

Les instructions permettant de récupérer des informations sur une base (telles que la liste de bases sur le serveur, le nom et la structure des tables) varient fortement d'une base à l'autre.

MySQL

SHOW DATABASES

Retourne la liste des noms des bases sur le serveur de bases de données.

Syntaxe	`SHOW DATABASES [LIKE nombase]`
`LIKE nombase`	Retourne la liste des bases ayant un nom correspondant à la chaîne `nombase` comportant des jokers `%` ou _.

SHOW TABLES

Retourne la liste des tables contenues dans la base.

Syntaxe	`SHOW TABLES [FROM nombase] [LIKE nomtable]`
`FROM nombase`	Retourne la liste des tables contenues dans la base `nombase`. Par défaut, ce sont les tables de la base actuellement connectée qui sont retournées.
`LIKE nomtable`	Retourne la liste des tables ayant un nom correspondant à la chaîne `nomtable` comportant des jokers `%` et _.

SHOW COLUMNS

Retourne la liste des champs contenus dans une table.

Syntaxe	`SHOW COLUMNS FROM nomtable [FROM nombase] [LIKE nomchamp]`
`nomtable`	Retourne la liste des champs de la table *nomtable*.
`FROM nombase`	Utilise la table *nomtable* de la base `nombase`. Par défaut, c'est la table de la base actuellement connectée qui est utilisée.
`LIKE nomchamp`	Retourne la liste des champs ayant un nom correspondant à la chaîne `nomchamp` comportant des jokers `%` et `_`.


REMARQUE

Describe

Il existe un équivalent à `SHOW COLUMNS FROM nomtable` *qui est* `DESCRIBE nomtable`.

10.5. Accéder à une base de données via PHP

Introduction

Dans le cas de l'utilisation d'une base de données via PHP, le client (on parle dans ce cas d'API) de la base de données est intégré (principalement via des bibliothèques `C`) au serveur web. Chaque base de données ayant son propre client, les noms des fonctions PHP diffèrent d'une base de données à l'autre. C'est pourquoi nous allons, serveur de bases de données après serveur de bases de données, décrire les fonctions qui leur sont propres. Nous vous invitons (après avoir fini la lecture de ce chapitre) à aller directement aux chapitres correspondant aux bases que vous serez amené à utiliser.

Principe général

Quoi qu'il en soit, le principe d'utilisation d'une base de données avec PHP reste le même d'une base à l'autre.

La première opération consiste à se connecter à la base de données. Cette opération se réalise en précisant l'adresse du serveur de la base de données (incluant éventuellement le port de communication), le nom de la base de données et, probablement, un nom d'utilisateur et un mot de passe.

Une fois connecté, vous récupérez un identifiant de session (une ressource) valable jusqu'à la déconnexion ou la fin d'exécution du script en cours. Cet identifiant de session, qui, selon les cas, doit être passé en paramètre des fonctions appelées ou est implicitement utilisé, vous permet alors d'exécuter des requêtes SQL.

Une fois la requête exécutée, vous récupérez un identifiant de résultat de requête (une ressource). Cet identifiant vous permet d'en lire le contenu (s'il s'agit d'une requête de type `SELECT`), généralement ligne par ligne.

Une fois toutes ces opérations réalisées, vous n'avez plus qu'à clore la connexion.

Connexion persistante

Comme vous venez de le découvrir, vous devrez ouvrir une connexion pour chaque script exécuté. Il n'est pas question d'ouvrir une connexion lors de l'exécution d'un script et de communiquer l'identifiant de connexion (ou même l'identifiant de résultat) au script suivant (afin, par exemple, de lire les résultats de la requête lancée par le premier script). Cela ne fonctionnerait pas du tout.

Même si ceci n'est pas une grosse contrainte en terme de programmation, cela peut le devenir en terme de performance. Si vous êtes confronté à un serveur de bases de données ayant un délai de connexion élevé, vous serez vite pénalisé. Pour pallier cet éventuel problème, PHP propose d'établir des connexions persistantes.

Qu'est-ce qu'une connexion persistante ? Globalement, il s'agit d'une connexion ouvrant une session qui pourra être utilisée par d'autres scripts que celui qui l'a initialisée. Cependant, le comportement d'une connexion persistante n'est pas exactement celui auquel vous auriez pu penser de prime abord.

En effet, si un utilisateur visite votre site dans lequel un script A ouvre une connexion persistante, il serait faux de penser que, nécessairement, lorsque l'utilisateur visitera le script B, il utilisera le même identifiant de session.

Pour bien comprendre cela, il faut revenir sur le principe de fonctionnement d'un serveur web (en tout cas, celui du serveur Apache). Un serveur web, ce n'est ni plus ni moins qu'un ensemble de processus (par défaut initialement dix sur un serveur Apache, le nombre pouvant augmenter suivant la demande) qui attendent qu'on leur demande une page. Ainsi, lorsqu'un utilisateur demande à visualiser une page, c'est le premier processus disponible qui s'occupe de répondre à la demande de son client. Donc, si l'utilisateur demande une page A, qui est en fait un script PHP ouvrant une connexion persistante, c'est ce premier processus qui va ouvrir et être détenteur de la session. Lorsque ce même utilisateur va demander une page B (un autre script PHP utilisant une connexion persistante), ce n'est pas nécessairement le processus précédent (peut-être est-il occupé avec un autre client) qui va répondre à la demande. Si ce nouveau processus n'a pas lui-même déjà eu l'occasion d'ouvrir une connexion persistante, il devra alors en ouvrir une. Dans le cas contraire, il n'aura pas besoin d'ouvrir une nouvelle connexion, mais il proposera une session différente de celle qui avait été ouverte avec le processus précédent.

Ainsi donc, si la connexion est bien persistante au niveau des processus du serveur, elle n'est pas pour autant véritablement maintenue tout au long du parcours d'un visiteur, puisqu'elle passe d'un visiteur à l'autre.

Il est à noter qu'à l'inverse des connexions classiques, les connexions persistantes ne peuvent être closes.


REMARQUE

Le danger des connexions non persistantes

Les connexions non persistantes sont censées être automatiquement closes à la fin de l'exécution d'un script. Mais, visiblement, il ne faut pas trop compter là-dessus. Sans que nous puissions véritablement l'affirmer, il semblerait que cela ne soit pas toujours vrai (un bug au niveau du système chargé de libérer les ressources ?) ou bien, plutôt, que cela ne soit pas immédiat. De la sorte, si, dans un script, vous omettez de clore une connexion, le nombre de sessions risque d'augmenter dramatiquement jusqu'à ce que la base de données rejette toute connexion. Nous vous conseillons donc de


REMARQUE

n'utiliser que des connexions persistantes, au moins dans cette configuration ; alors le nombre de connexions reste limité par le nombre de processus du serveur web.

Couches d'abstraction

Comme les fonctionnalités proposées d'une base de données à l'autre sont assez similaires, des efforts sont faits pour créer des fonctions pouvant être utilisées par le plus grand nombre de serveurs de bases de données possible. On parle alors de couche d'abstraction.

ODBC

À mi-chemin entre les deux, nous trouvons les serveurs de bases de données qui utilisent un protocole unifié, en l'occurrence ODBC. Pour l'ensemble de ces bases, vous pourrez utiliser les mêmes fonctions (celles qui commencent par `odbc`).

10.6. Présentation de l'application d'exemple

Pour l'ensemble des bases de données évoquées dans ce chapitre, la description des fonctions offertes par PHP sera complétée par deux exemples. Le premier est un compteur de cliques. Cet exemple étant très simple il sera décliné dans chaque chapitre. Le second, quant à lui, est plus complet, il s'agit en effet d'une application de type bibliothèque/filmothèque/discothèque/photothèque que nous appellerons "superthèque". Dans ce cas, nous vous présenterons ici le code commun et nous mettrons à profit les possibilités offertes par la programmation orientée objet (que nous vous avons fait découvrir dans les premiers chapitres de ce livre) pour n'avoir par la suite qu'à récrire que quelques classes ou méthodes afin de tenir compte de la spécificité de la base de données étudiée.

Ces exemples ont été choisis parce qu'ils correspondent à des besoins fréquemment rencontrés dans la conception d'un site web, et qu'ils répondent à certaines problématiques parfois soulevées en d'autres circonstances. L'intérêt de ces scripts va donc au-delà de la simple mise en œuvre d'une base de données.

Notre "superthèque" sera constituée d'une collection d'articles (livre, film, musique, etc.) regroupés en albums. Cet exemple est très intéressant, car il offre des solutions à de nombreux problèmes :

- Comment afficher N articles par page ?
- Comment modifier l'ordre d'affichage des articles ?
- Comment filtrer les articles ?
- Comment créer et gérer un formulaire de saisie ?
- Et bien d'autres points encore…

Ce sont ces mêmes problèmes que l'on rencontre dans le cas de la création d'un forum, d'un moteur de recherche, d'un site de petites annonces, etc.

REMARQUE

Principe de fonctionnement d'un moteur de recherche

Puisque nous abordons ce sujet, cela vous intéresse peut-être de mieux connaître le principe de fonctionnement d'un moteur de recherche. Il y a, en fait, deux parties distinctes : l'une (appelée spider) est chargée de collecter les informations (telle une araignée sur la toile) et l'autre (le moteur de recherche proprement dit) est chargée de restituer les informations selon les mots-clés saisis par l'utilisateur.

Le principe du spider est relativement simple. Il s'agit d'un logiciel qui lit le contenu d'une page web (quelconque), l'analyse (en extrait principalement le texte hors HTML), stocke dans une base de données l'URL de la page et le texte associé, puis stocke l'URL des pages proposées en lien pour pouvoir les analyser ultérieurement, et ainsi de suite.

La base du moteur de recherche est celle que nous avons utilisée ici. En pratique, les moteurs utilisent de savants algorithmes pour afficher les résultats par ordre de pertinence (généralement, sont considérés comme pertinents les sites vers lesquels pointent de nombreux autres sites).

Par défaut, l'application affichera la liste des articles et albums stockés dans la base de données. Pour des contraintes d'affichage, seul le titre et le type de l'article seront affichés, de plus un maximum de 10 articles (valeur paramétrable) sera disponible par page. Par conséquent, le cas échéant un lien sera proposé pour accéder à la page de résultats suivante ou précédente. L'information détaillée d'un article sera disponible par un lien affiché à coté du résumé de l'article dans la liste. Par commodité, nous proposerons d'afficher la liste des articles triés par titre ou type, au choix soit dans l'ordre croissant soit dans l'ordre décroissant. Nous offrirons également la possibilité de filtrer les articles par titre et/ou par type.

Bref, l'application aura l'allure suivante:

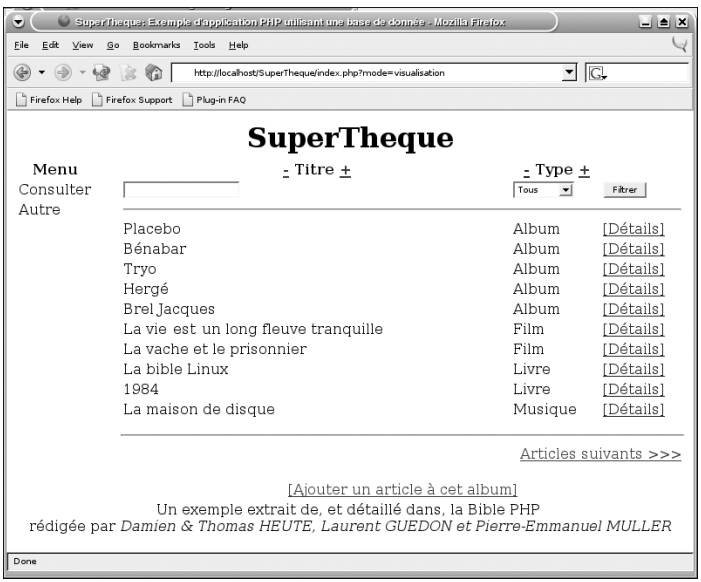


Figure 10.1 : *SuperTheque*

et offrira la possibilité de voir le détail d'un article

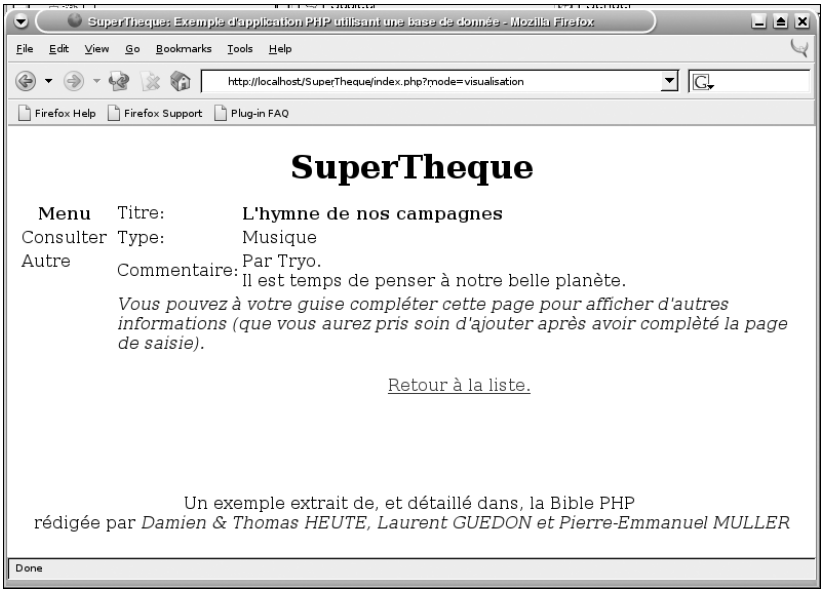


Figure 10.2 : *SuperTheque*

Il sera bien évidemment possible d'ajouter un article à l'album sélectionné ; pour cela l'utilisateur aura la possibilité de saisir un titre (obligatoire) et un commentaire et devra sélectionner le type de l'article comme le montre l'écran suivant:

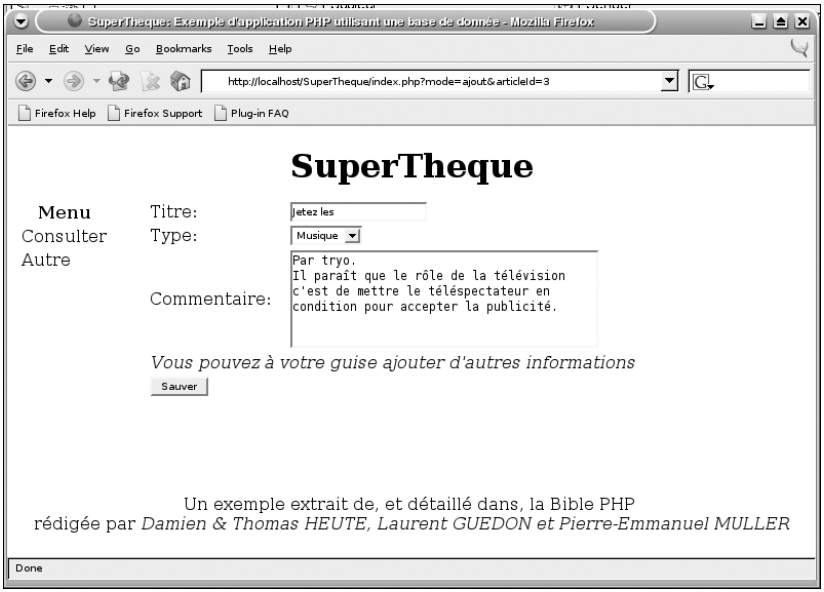


Figure 10.3 : *SuperTheque*

Certes la présentation est largement perfectible mais dans l'immédiat ce sont les fonctionnalités qui nous intéressent.

Le modèle de données

Le modèle de données se déduit naturellement de la présentation qui vient d'être faite de l'application.

L'objet principal est bien entendu l'article. Celui-ci porte au minimum un titre et éventuellement un commentaire et est d'un type donné : livre, film, musique, photo, etc. Nous ajouterons un type particulier baptisé "album" qui aura pour objet de rassembler une collection d'articles. Par conséquent, chaque article (y compris un album) appartiendra à un album donné.

L'objet de base sera donc l'objet `Article` dont voici le code:

Listing 10.1 : Article_class.php

```
<?php
/**
 * Article.php
 * Objet representant un objet quelconque de la collection.
 * Compatibilite: PHP 5
 */
class Article
{
    // Identifiant unique de l'article
    protected $id;
    // Titre de l'article
    protected $titre;
    // Identifiant du type de l'article
    protected $typeId;
    // Identifiant de l'album auquel appartient l'article
    protected $albumId;
    // Commentaire sur l'article
    protected $commentaire;

    public function __construct($id=-1, $albumId=-1, $titre="", $typeId=-1)
    {
        $this->setId($id);
        $this->setTitre($titre);
        $this->setTypeId($typeId);
        $this->setAlbumId($albumId);
    }

    public function getId()
    {
        return $this->id;
    }

    public function setId($id)
    {
```

```
        $this->id = $id;
    }

    public function getTitre()
    {
        return $this->titre;
    }

    public function setTitre($titre)
    {
        $this->titre = $titre;
    }

    public function getAlbumId()
    {
        return $this->albumId;
    }

    public function setAlbumId($albumld)
    {
        $this->albumId = $albumId;
    }

    public function getTypeId()
    {
        return $this->typeId;
    }

    public function setTypeId($typeId)
    {
        $this->typeId = $typeId;
    }

    public function getCommentaire()
    {
        return $this->commentaire;
    }

    public function setCommentaire($commentaire)
    {
        $this->commentaire = $commentaire;
    }
}
?>
```

Notre application n'ayant pas pour objectif d'afficher l'ensemble des articles mais seulement ceux de l'album sélectionné et éventuellement uniquement les articles ayant un titre et/ou un type donné, nous utiliserons un filtre matérialisé par l'objet `Filtre` suivant:

Listing 10.2 : Filtre_class.php

```
<?php
class Filtre
{
    private $albumId=-1;
    private $titre;
    private $typeId=-1;

    public function setAlbumId($albumId)
    {
        $this->albumId = $albumId;
    }

    public function getAlbumId()
    {
        return $this->albumId;
    }

    public function setTitre($titre)
    {
        $this->titre = $titre;
    }

    public function getTitre()
    {
        return $this->titre;
    }

    public function setTypeId($typeId)
    {
        if ($typeId == "") $typeId = -1;
        $this->typeId = $typeId;
    }

    public function getTypeId()
    {
        return $this->typeId;
    }

}
?>
```

De même il n'est pas question d'afficher tous les articles répondant aux critères précédents dans une seule et même page. Il faut donc en extraire une plage de résultat. À cet effet, nous utiliserons la classe `Plage` suivante:

Listing 10.3 : Plage_class.php

```
<?php
class Plage
```

```
{
    private $premierArticle;
    private $nbArticleMax;

    public function setPremierArticle($premierArticle)
    {
        $this->premierArticle = $premierArticle;
    }

    public function getPremierArticle()
    {
        return $this->premierArticle;
    }

    public function setNbArticleMax($nbArticleMax)
    {
        $this->nbArticleMax = $nbArticleMax;
    }

    public function getNbArticleMax()
    {
        return $this->nbArticleMax;
    }
}
?>
```

Nous utiliserons également un objet `Tri` afin de définir la manière dont doivent être triés les articles lors de l'affichage.

Listing 10.4 : Tri_class.php

```
<?php
class Tri
{
    private $champ;
    private $sens; // -1 Décroissant, 0 sans, 1 Croissant

    public function setChamp($champ)
    {
        $this->champ = $champ;
    }

    public function getChamp()
    {
        return $this->champ;
    }

    public function setSens($sens)
    {
        $this->sens = $sens;
    }
```

```
    public function getSens()
    {
        return $this->sens;
    }

}
?>
```


Schéma de la base de données

En base de données, nous utiliserons les tables suivantes:

Tableau 10.12 : Articles

Champ	Type	Commentaire
id	Entier auto-incrémenté	Identifiant unique de l'article
albumId	Entier non null	Identifiant de l'album (article de type album) auquel appartient l'article
titre	Chaîne de caractères	Titre de l'article
typeId	Entier non null	Identifiant du type de l'article (voir table types)
commentaire	Chaîne de caractères pouvant être null	

Tableau 10.13 : Types

Champ	Type	Commentaire
id	Entier auto incrémenté	Identifiant unique du type
type	Chaîne de caractères	Nom du type

Evitons les conflits

Si vous souhaitez diffuser vos scripts, vous êtes invité à faire précéder les noms des tables d'un préfixe librement configurable afin que l'utilisateur puisse éviter tout conflit (simplement en changeant le préfixe) entre les noms des tables de vos scripts et ceux d'autres scripts dans le cas où ils devraient utiliser la même base.

Le contrôleur

Le cœur de l'application, son moteur, que l'on appelle aussi le contrôleur, doit permettre de gérer les différentes interactions de l'utilisateur.

Une des premières opérations consiste à lire le fichier de configuration

Listing 10.5 : index.php (extrait)

```
include_once("config/supertheque_cfg.php");
```

dont voici le maigre contenu

Listing 10.6 : supertheque_cfg.php

```
<?php
$nbArticleMax   = 10;

// Decommenter la ligne adequat
//-----------------------------
//$ressource = "MySQL";
$ressource = "SQLite";
//$ressource = "MSSQLServer";
//$ressource = "Demo";
?>
```

Ce fichier permet de préciser combien d'articles seront affichés par page et quelle base de données (ou d'une manière générale quelle ressource) sera utilisée.

Avant de pouvoir commencer le contrôleur doit connaître l'état courant de l'application: à savoir quel est l'article qui a été sélectionné? quels sont les critères de filtrage qui ont été choisis? etc. Pour cela, et par soucis de simplicité, ces différents paramètres seront stockés et lus en session.

Pour chacun de ces paramètres nous vérifions donc s'il existe en session (via la fonction `isset()`) et s'il n'est pas disponible nous initialisons sa valeur avant de le stocker en session.

Listing 10.7 : index.php (extrait)

```
// Lecture des informations stockee en session
// ou initialisation si necessaire
//--------------------------------------------
if (isset($_SESSION["filtre"])) {
    $filtre = $_SESSION["filtre"];
} else {
    $filtre = new Filtre();
    $filtre->setAlbumId(0);
    $filtre->setTitre("");
    $filtre->setTypeId(-1);
    $_SESSION["filtre"] = $filtre;
}
if (isset($_SESSION["plage"])) {
    $plage = $_SESSION["plage"];
} else {
    $plage = new Plage();
    $plage->setPremierArticle(0);
    $plage->setNbArticleMax($nbArticleMax);
    $_SESSION["plage"] = $plage;
}
```

```
if (isset($_SESSION["tri"])) {
    $tri = $_SESSION["tri"];
} else {
    $tri = new Tri();
    $tri->setChamp("titre");
    $tri->setSens(1);
    $_SESSION["tri"] = $tri;
}
```

Attention, car avant de pouvoir manipuler les variables de session, il faut faire appel à `session_start()`. Plus encore, il est impératif de déclarer les classes utilisées dans les variables de session avant cet appel à `session_start()`. Nous insérerons donc avant le code précédent, le code:

Listing 10.8 : index.php (extrait)

```
include_once("classes/Article_class.php");
include_once("classes/Filtre_class.php");
include_once("classes/Tri_class.php");
include_once("classes/Plage_class.php");
include_once("classes/SuperTheque_class.php");
include_once("config/supertheque_cfg.php");
include_once("classes/Ressource".$ressource."_class.php");

session_start();
```

Nous retrouvons ici, le fichier de configuration évoqué précédemment ainsi que deux nouveaux fichiers de classes que nous verrons plus loin. Notez que le chargement de la dernière classe dépend du paramètre de configuration `$ressource`.

Vous pouvez vous reporter aux sections "Les inclusions de fichiers" et "Les sessions" pour plus de détails.

Le contrôleur travaillera selon le principe que chaque action de l'utilisateur est identifiée par un paramètre `mode`. Ce paramètre pourra être passé soit au travers de l'URL appelée avec un lien selon le modèle `<a href="?mode=modeselectionné">`, soit au travers d'un formulaire avec un champ caché `<input type="hidden" name="mode" value="modeselectionné" />`. La lecture de ce mode se fera donc ainsi:

Listing 10.9 : index.php (extrait)

```
if (isset($_POST["mode"])) {
   $mode = $_POST["mode"];
} else if (isset($_GET["mode"])) {
   $mode = $_GET["mode"];
} else {
   $mode = "visualisation";
}
```


RENVOI

Vous pouvez vous reporter à la sections "Les variables" pour plus de détails.

Nous avons identifié les actions suivantes:

- article: afin de sélectionner un article (ou album)
- tri: afin de trier les articles selon un champ (titre ou type) et un sens donné (croissant, décroissant)
- suivants, précédents: afin d'afficher les N articles suivants ou précédents
- filtre: afin de définir un nouveau filtre (basé sur un titre ou type)
- visualisation: il s'agit du mode par défaut
- ajout: afin de proposer un formulaire pour l'ajout d'un article
- sauver: afin de sauvegarder les données soumises via le formulaire d'ajout d'un article

Mode = article

Lorsque nous voudrons proposer à l'utilisateur de sélectionner un article (ou album) donné, nous n'aurons qu'à insérer un lien du type `<a href="?mode=article &articleId=<articleId>">` (ou à utiliser un champ caché dans le cas d'un formulaire en mode `post`). Le contrôleur doit alors modifier les critères de filtrage pour préciser quel est le nouvel article (ou album) sélectionné.

Attention, il y a toutefois une subtilité: Si l'utilisateur sélectionne un album dans la cinquième page de la liste du contenu de l'album père, il faut prendre soin de ne pas sélectionner la cinquième page de l'album fils car celui-ci contient peut-être moins de cinq pages. Il faut donc revenir à la première page de l'album sélectionné. C'est ce qui est fait ici.

Listing 10.10 : index.php (extrait)

```
if ($mode == "article") {
    //----------------------------------------
    // Quel article (ou album) devra etre affiche ?
    //----------------------------------------
    if (isset($_POST["articleId"])) {
       $articleId = $_POST["articleId"];
    } else if (isset($_GET["articleId"])) {
       $articleId = $_GET["articleId"];
    } else {
       $articleId = 0;
    }
    // Modification des parametres de filtrage
    $album = $supertheque->getArticle($articleId);
    $filtre->setAlbumId($articleId);
    $_SESSION["filtre"] = $filtre;

    // Si l'on passe d'un album a un autre
```

```
    // mieux vaut aller au premier article de l'album
    if ($supertheque->isAlbum($album)) {
        $plage->setPremierArticle(0);
        $_SESSION["plage"] = $plage;
    }
    // Passage en mode visualisation (avec le nouveau filtre)
    header("Location: ?mode=visualisation");
    die();
}
```

Mode = tri

Si l'utilisateur clique sur un lien de la forme `"?mode=tri&tri_champ=titre&tri_sens=1"` alors l'action à déclencher consiste uniquement à modifier les paramètres de tri (champ sur lequel s'applique le tri et sens du tri : -1 si décroissant, 0 si non trié, 1 si croissant) stockés en session et retourner à la page principale.

Listing 10.11 : index.php (extrait)

```
if ($mode == "tri") {
    // Modification des paramètres de tri
    $tri->setChamp($_GET["tri_champ"]);
    $tri->setSens($_GET["tri_sens"]);
    $_SESSION["tri"] = $tri;
    // Passage en mode visualisation (avec le nouveau tri)
    header("Location: ?mode=visualisation");
    die();
}
```

Mode = suivants, Mode = precedents

Si l'utilisateur clique sur un lien de la forme `"?mode=suivants" ou "?mode=precedents"` alors l'action à déclencher consiste uniquement à modifier, en session, les paramètres définissant la plage d'articles (en fait uniquement l'index du premier article puisque le nombre d'articles à afficher fait quant à lui des paramètres de configuration du script) à afficher et retourner à la page principale.

Listing 10.12 : index.php (extrait)

```
if ($mode == "suivants") {
    // Modification de la plage de selection
    $plage->setPremierArticle($plage->getPremierArticle() +
                              $nbArticleMax);
    $_SESSION["plage"] = $plage;
    // Passage en mode visualisation
    header("Location: ?mode=visualisation");
    die();
}

if ($mode == "precedents") {
```

```
    // Modification de la plage de selection
    $plage->setPremierArticle($plage->getPremierArticle() -
                              $nbArticleMax);
    $_SESSION["plage"] = $plage;
    // Passage en mode visualisation
    header("Location: ?mode=visualisation");
    die();
}
```

Mode = filtre

Si l'utilisateur clique sur le bouton "filtrer" alors un formulaire est soumis avec l'attribut caché "mode" positionné à la valeur "filtre". Il convient alors de modifier les paramètres de filtrage puis retourner à la page principale.

Listing 10.13 : index.php (extrait)

```
if ($mode == "filtre") {
    // Modification des parametres de filtrage
    $filtre->setTitre($_POST["filtre_titre"]);
    $filtre->setTypeId($_POST["filtre_typeId"]);
    $_SESSION["filtre"] = $filtre;
    // Passage en mode visualisation (avec le nouveau filtre)
    header("Location: ?mode=visualisation");
    die();
}
```

Mode = visualisation

Par défaut la page principale affiche la liste des articles correspondants aux critères de sélection (le filtre), triés selon la règle stockée en session et en limitant l'affichage à la plage d'articles sélectionnés (elle aussi stockée en session). Pour cela l'application doit accéder à la base de données ou d'une manière générale à une ressource quelconque (une zone mémoire, un fichier par exemple). Cette ressource devra nous permettre de: nous connecter via une méthode `connexion()`, nous déconnecter via une méthode `deconnexion()`, récupérer la liste des articles via une méthode `getArticles()`, déterminer le nombre d'articles répondant aux critères de sélection via une méthode `getNbTotalArticles()` (pour éventuellement proposer un lien "page suivante"), récupérer la liste des types connus via une méthode `getTypes()` (pour la sélection du filtre). Afin de déterminer si un article donné est ou non un album, cette ressource devra également nous retourner l'identifiant du type "album" via une méthode `getAlbumTypeId()`. Et comme nous le verrons bientôt, nous aurons besoin d'une méthode permettant l'ajout d'un article dans la base via la méthode `addArticle()`.

En quelques mots, nous avons dressé l'interface de l'objet qui nous permettra d'accéder à une ressource quelle qu'elle soit (Un fichier, une base de données MySQL, Oracle, etc.). D'où l'objet `RessourceInterface`:

Listing 10.14 : RessourceInterface_class.php

```
<?php
/**
 * RessouceInterface_class.php
 */
interface RessourceInterface
{
    /**
     * Retourne l'identifiant de connexion a la base de donnees
     */
    public function connexion();

    /**
     * Demande la deconnexion a la base de donnees
     */
    public function deconnexion();

    /**
     * Re-cree la structure de la base de donnees
     */
    public function reset();

    /**
     * Ajoute un article en base
     * avec les caracteristiques donnees en parametre.
     * REM: Nous aurions egalement pu (du ?) proposer
     * une interface avec comme parametre un objet Article
     */
    public function addArticle($albumId,
                               $titre,
                               $typeId,
                               $commentaire);
    /**
     * Retourne l'objet Article correspondant a l'article
     * identifie par articleId
     */
    public function getArticle($articleId);

    /**
     * Retourne un tableau indexe des articles repondant
     * aux criteres de filtrage, trie et dont la plage nous interessant
     * a ete extraite
     */
    public function getArticles(Filtre $filtre, Plage $plage, Tri $tri);

    /**
     * Retourne le nombre total d'articles repondant
     * aux criteres de filtrage
     */
    public function getNbTotalArticles(Filtre $filtre);
```

```
    /**
     * Retourne le tableau associatif des types
     * ou la cle est l'identifiant du type
     *    la valeur le nom du type
     */
    public function getTypes();

    /**
     * Retourne l'identifiant du type associé au type album
     */
    public function getAlbumTypeId();
}
?>
```

En fait, nous n'attaquerons pas la classe `Ressource` directement puisque nous passerons par une classe `SuperTheque` chargée d'effectuer des tests et traitements complémentaires (même si en l'occurrence, ici, nous aurions sans doute pu nous en passer).

Listing 10.15 : SuperTheque_class.php

```
<?php
/**
 * SuperTheque_class.php
 * Objet permettant d'acceder a la discoTheque, filmoTheque, etc.
 * a priori stockee dans une base de donnees.
 */
class SuperTheque
{
    /**
     * Objet ressource utilise pour acceder aux articles.
     * Ce sera un objet pour recuperer les articles dans
     * une base de donnees mais cela pourrait tout aussi
     * bien etre un objet pour recuperer les articles dans
     * un fichier ou autres..
     */
    protected $ressource;

    public function __construct($ressource)
    {
        $this->ressource = $ressource;
    }

    public function reset()
    {
        $this->ressource->connexion();
        $articles = $this->ressource->reset();
        $this->ressource->deconnexion();
    }

    public function addArticle($albumId,
                               $titre,
                               $typeId,
```

```
                            $commentaire)
{
    $this->ressource->connexion();
    $articles = $this->ressource->addArticle($albumId,
                                             $titre,
                                             $typeId,
                                             $commentaire);
    $this->ressource->deconnexion();
}

public function getArticle($articleId)
{
    // L'article racine (0) n'est pas un "vrai" article
    // il ne doit pas être récupéré en BD.
    if (($articleId == "")||($articleId == 0)) {
        return new Article(0, -1, "", $this->getAlbumTypeId());
    }

    $this->ressource->connexion();
    $articles = $this->ressource->getArticle($articleId);
    $this->ressource->deconnexion();

    return $articles;
}

/**
 * Retourne les N articles du type selectionne
 * a partir du I-ieme, lorsqu'ils sont classes
 * dans l'ordre precise
 * @param filtre Filtre de selection
 * @param plage Plage des enregistrements a retourner
 * @param tri Regle de tri
 */
public function getArticles(Filtre $filtre, Plage $plage, Tri $tri)
{
    $this->ressource->connexion();
    $articles = $this->ressource->getArticles($filtre, $plage, $tri);
    $this->ressource->deconnexion();

    return $articles;
}

public function getNbTotalArticles(Filtre $filtre)
{
    $this->ressource->connexion();
    $articles = $this->ressource->getNbTotalArticles($filtre);
    $this->ressource->deconnexion();

    return $articles;
}

public function getTypes()
```

```
    {
        $this->ressource->connexion();
        $types = $this->ressource->getTypes();
        $this->ressource->deconnexion();

        return $types;
     }

     public function getAlbumTypeId()
     {
         return $this->ressource->getAlbumTypeId();
     }

     public function isAlbum($article)
     {
         if ($article->getTypeId() == $this->getAlbumTypeId()) return TRUE;
         else return FALSE;
     }
}
?>
```

L'objet `supertheque` sera donc instancié comme suit:

Listing 10.16 : index.php (extrait)

```
$supertheque = new SuperTheque(new Ressource());
```

Pour préparer l'affichage de la liste des articles d'un album ou du détail d'un article nous devons dans un premier temps utiliser cet objet `supertheque` pour récupérer les données relatives à l'article (ou album) sélectionné. Ce que nous stockerons dans une variable `$article`.

S'il s'agit d'un album nous devons récupérer l'ensemble des articles que nous stockerons dans un tableau `$articles`.

Nous verrons, plus loin, qu'il est indispensable de connaître le nombre total d'articles `$nbTotalArticles` répondant aux critères du filtre afin d'afficher ou non un lien permettant d'accéder à la page suivante.

Listing 10.17 : index.php (extrait)

```
if ($mode == "visualisation") {
    // Visualiser l'article (ou album) specifie (ou celui par défaut)
    $article = $supertheque->getArticle($filtre->getAlbumId());
    if ($supertheque->isAlbum($article)) {
        $articles = $supertheque->getArticles($filtre,
                                              $plage,
                                              $tri);
        $nbTotalArticles = $supertheque->getNbTotalArticles($filtre);
    }
}
```

Voilà, nous disposons de toutes les informations nécessaires à l'affichage des articles d'un album ou du détail d'un article. Selon le cas nous passerons donc à l'une ou l'autre des vues

Listing 10.18 : index.php (extrait)

```
if ($mode == "visualisation")
{
    if ($supertheque->isAlbum($article)) {
        include_once("vues/SuperTheque_VueVisualisation_inc.php");
    } else {
        include_once("vues/SuperTheque_VueDetail_inc.php");
    }
}
```

Nous sommes quasiment prêts à passer aux aspects relatifs au rendu visuel de notre application. Reste toutefois le cas de l'ajout d'un article dans la base.

Mode = ajout

L'application proposera un lien permettant de passer du mode visualisation au mode ajout dans lequel l'affichage des articles est remplacé par l'affichage d'un formulaire permettant la saisie des données relatives au nouvel article.

Ce mode se résume (du point de vue du contrôleur) donc à :

Listing 10.19 : index.php (extrait)

```
if ($mode == "ajout")
{
   include_once("vues/SuperTheque_VueAjout_inc.php");
}
```

Mode = sauver

Une fois les données du formulaire soumises, il faut les récupérer et les valider.

Nous aurons un champ caché précisant à quel album doit être rattaché le nouvel article. Le champ titre est obligatoire. A chaque erreur détectée un tableau `$erreurs` sera alimenté. Si aucune erreur n'a été détectée alors il sera possible d'ajouter l'article dans la base, sinon la page contenant le formulaire sera affiché de nouveau avec les messages d'erreur simplement en passant du mode sauver au mode ajout.

Listing 10.20 : index.php (extrait)

```
if ($mode == "sauver") {
    // Ajouter le nouvel article a l'album $albumId
    $albumId = $_POST["albumId"];
    // Verifions tout de même que l'article $albumId est bien un album
    $album = $supertheque->getArticle($albumId);
    if (!$supertheque->isAlbum($album)) {
```

```
            $erreurs[] = "L'article s&eactue;l&eacute;ctionn&eacute; n'est pas".
                         " un album!";
        }
        // Verifions que le champ titre a été sélectionné
        $titre = $_POST["titre"];
        if ((!isset($titre))||(strlen($titre)==0)) {
            $erreurs[] = "Le titre ne doit pas &ecirc;tre vide.";
        }
        $typeId = $_POST["typeId"];

        if (count($erreurs)>0)
        {
            // Au moins une erreur a ete detectee alors on ne
            // sauvegarde plus, on revient a la page d'ajout.
            $mode = "ajout";
        } else {
            $supertheque->addArticle($albumId,
                                stripSlashes($_POST["titre"]),
                                $_POST["typeId"],
                                stripSlashes($_POST["commentaire"]));
            // Une fois sauvegarde on repasse en mode visualisation
            header("Location: ?mode=visualisation");
            die();
        }
}
```

Ce bout de code amène à une remarque très importante liée à l'utilisation de `stripSlashes()`.

Magic quotes

Par défaut, PHP est configuré avec l'option *magic_quotes* activée, ce qui signifie que les valeurs passées par formulaire sont traitées pour ajouter un anti-slash devant les apostrophes. Il faut donc en tenir compte (afin de le supprimer), notamment, ici, lors de l'appel à la méthode d'ajout d'un article pour les champs de type texte (titre et commentaire). La valeur passée à la méthode `addArticle()` n'est donc pas `$_POST["commentaire"]`, mais `stripSlashes($_POST["commentaire"])`.

La totale

Voici donc l'implémentation complète du contrôleur:

Listing 10.21 : index.php

```
<?php
include_once("classes/Article_class.php");
include_once("classes/Filtre_class.php");
include_once("classes/Tri_class.php");
include_once("classes/Plage_class.php");
include_once("classes/SuperTheque_class.php");
include_once("config/supertheque_cfg.php");
include_once("classes/Ressource".$ressource."_class.php");
```

```
session_start();

$supertheque = new SuperTheque(new Ressource());

// Lecture des informations stockee en session
// ou initialisation si necessaire
//-------------------------------------------
if (isset($_SESSION["filtre"])) {
    $filtre = $_SESSION["filtre"];
} else {
    $filtre = new Filtre();
    $filtre->setAlbumId(0);
    $filtre->setTitre("");
    $filtre->setTypeId(-1);
    $_SESSION["filtre"] = $filtre;
}
if (isset($_SESSION["plage"])) {
    $plage = $_SESSION["plage"];
} else {
    $plage = new Plage();
    $plage->setPremierArticle(0);
    $plage->setNbArticleMax($nbArticleMax);
    $_SESSION["plage"] = $plage;
}
if (isset($_SESSION["tri"])) {

    $tri = $_SESSION["tri"];
} else {
    $tri = new Tri();
    $tri->setChamp("titre");
    $tri->setSens(1);
    $_SESSION["tri"] = $tri;
}


//----------------------------------------
// Que veux le visiteur ?
//----------------------------------------
if (isset($_POST["mode"])) {
   $mode = $_POST["mode"];
} else if (isset($_GET["mode"])) {
   $mode = $_GET["mode"];
} else {
   $mode = "visualisation";
}

if ($mode == "article") {
    //----------------------------------------
    // Quel article (ou album) devra etre affiche ?
    //----------------------------------------
    if (isset($_POST["articleId"])) {
```

```
        $articleId = $_POST["articleId"];
    } else if (isset($_GET["articleId"])) {
        $articleId = $_GET["articleId"];
    } else {
        $articleId = 0;
    }
    // Modification des parametres de filtrage
    $album = $supertheque->getArticle($articleId);
    $filtre->setAlbumId($articleId);
    $_SESSION["filtre"] = $filtre;

    // Si l'on passe d'un album a un autre
    // mieux vaut aller au premier article de l'album
    if ($supertheque->isAlbum($album)) {
        $plage->setPremierArticle(0);
        $_SESSION["plage"] = $plage;
    }
    // Passage en mode visualisation (avec le nouveau filtre)
    header("Location: ?mode=visualisation");
    die();
}

if ($mode == "filtre") {
    // Modification des parametres de filtrage
    $filtre->setTitre($_POST["filtre_titre"]);
    $filtre->setTypeId($_POST["filtre_typeId"]);
    $_SESSION["filtre"] = $filtre;
    // Passage en mode visualisation (avec le nouveau filtre)
    header("Location: ?mode=visualisation");
    die();
}

if ($mode == "tri") {
    // Modification des parametres de tri
    $tri->setChamp($_GET["tri_champ"]);
    $tri->setSens($_GET["tri_sens"]);
    $_SESSION["tri"] = $tri;
    // Passage en mode visualisation (avec le nouveau tri)
    header("Location: ?mode=visualisation");
    die();
}

if ($mode == "suivants") {
    // Modification de la plage de selection
    $plage->setPremierArticle($plage->getPremierArticle() +
                              $nbArticleMax);
    $_SESSION["plage"] = $plage;
    // Passage en mode visualisation
    header("Location: ?mode=visualisation");
    die();
}
```

```
if ($mode == "precedents") {
    // Modification de la plage de selection
    $plage->setPremierArticle($plage->getPremierArticle() -
                              $nbArticleMax);
    $_SESSION["plage"] = $plage;
    // Passage en mode visualisation
    header("Location: ?mode=visualisation");
    die();
}


if ($mode == "sauver") {
    // Ajouter le nouvel article a l'album $albumId
    $albumId = $_POST["albumId"];
    // Verifions tout de même que l'article $albumId est bien un album
    $album = $supertheque->getArticle($albumId);
    if (!$supertheque->isAlbum($album)) {
       $erreurs[] = "L'article s&eactue;l&eacute;ctionn&eacute; n'est pas".
                    " un album!";
    }
    // Verifions que le champ titre a été sélectionné
    $titre = $_POST["titre"];
    if ((!isset($titre))||(strlen($titre)==0)) {
        $erreurs[] = "Le titre ne doit pas &ecirc;tre vide.";
    }
    $typeId = $_POST["typeId"];

    if (count($erreurs)>0)
    {
        // Au moins une erreur a ete detectee alors on ne
        // sauvegarde plus, on revient a la page d'ajout.
        $mode = "ajout";
    } else {
        $supertheque->addArticle($albumId,
                             stripSlashes($_POST["titre"]),
                             $_POST["typeId"],
                             stripSlashes($_POST["commentaire"]));
        // Une fois sauvegarde on repasse en mode visualisation
        header("Location: ?mode=visualisation");
        die();
    }
}

if ($mode == "visualisation") {
    // Visualiser l'article (ou album) specifie (ou celui par défaut)
    $article = $supertheque->getArticle($filtre->getAlbumId());
    if ($supertheque->isAlbum($article)) {
        // S'il s'agit d'un album voici ce que nous devons faire

        //----------------------
        // Creation du Filtre
        //----------------------
```

```
        $articles = $supertheque->getArticles($filtre,
                                              $plage,
                                              $tri);
        $nbTotalArticles = $supertheque->getNbTotalArticles($filtre);
    }
}

// Pour l'affichage, nous aurons besoin de connaitre les noms
// des differents types existants
$types = $supertheque->getTypes();

//-----------------
// Affichage
//-----------------
include_once("entete_inc.php");
if ($mode == "visualisation")
{
    if ($supertheque->isAlbum($article)) {
        include_once("vues/SuperTheque_VueVisualisation_inc.php");
    } else {
        include_once("vues/SuperTheque_VueDetail_inc.php");
    }
} else if ($mode == "ajout")
{
   include_once("vues/SuperTheque_VueAjout_inc.php");
}

include_once("pieddepage_inc.php");
?>
```

Vous y retrouvez l'ensemble des lignes de code précédent. Les seules lignes passées sous silence jusque là, sont celles permettant de récupérer dans un tableau associatif `$types` les noms des différents types (d'article) disponibles.

Listing 10.22 : index.php (extrait)

```
// Pour l'affichage, nous aurons besoin de connaitre les noms
// des differents types existants
$types = $supertheque->getTypes();
```

Et celles relatives à la mise en page

```
include_once("entete_inc.php");
```

et

```
include_once("pieddepage_inc.php");
```

que nous aborderons dès le chapitre suivant.

Les vues

Une fois le traitement opéré (c'est à dire, une fois les données mises à jour ou récupérées) il est possible de passer à l'affichage de la vue.

Comme nous l'avons vu, cette application dispose de trois écrans différents :

- L'affichage de la liste des articles (mode visualisation d'un album)
- L'affichage des détails d'un article (mode visualisation d'un article)
- La page d'ajout d'un nouvel article.

Toutefois, toutes trois auront globalement la même mise en page. Nous voulons dire par-là, quelles auront toutes le même bandeau supérieur, le même menu gauche et ou droit, et éventuellement un bandeau inférieur. Afin d'éviter de dupliquer du code, il suffit de créer un fichier que nous appellerons `entete_inc.php` décrivant le bandeau supérieur et éventuellement le menu gauche et un autre décrivant le menu droit et éventuellement le bandeau inférieur appelé `pieddepage_inc.php`

Le squelette du fichier d'en-tête est le suivant :

Listing 10.23 : entete_inc.php (squelette)

```
<html>
<body>
<table>
<tr><td colspan="3"><!-- Entete proprement dit --></td></tr>
<tr><td><!-- Menu Gauche--></td><td>
```

alors que le pied de page a l'allure suivante :

Listing 10.24 : pieddepage_inc.php (squelette)

```
</td><td><!-- Menu droit --></td></tr>
<tr><td colspan="3"><!-- Pied de page proprement dit --></td></tr>
</body>
</html>
```

Il n'y a plus qu'à mettre la page principale entre ces deux fichiers. C'est ce que le contrôleur réalise en effectuant les différents `include` relatifs à la vue.

Vue liste des articles

Pour rappel, à l'issue du traitement du contrôleur nous disposons des variables suivantes:

`$article`, qui contient les informations relatives à l'article sélectionné.

`$articles`, qui contient la liste des articles (filtrés, triés et extraits) de l'album sélectionné

`$nbTotalArticles`, qui contient le nombre total d'articles répondant aux critères de filtrage.

`$types`, qui contient la liste des types disponibles.

mais aussi à tout moment de `$plage`, `$filtre` et `$tri`.

À la base, ce qu'il nous faut c'est afficher le contenu du tableau `$articles` ce qui se fait aisément via une boucle `foreach($articles as $articleListe)`. Mais il faut également:

- proposer les liens permettant de modifier l'ordre de tri
- intégrer un formulaire pour la saisie des critères de filtrage. Ce dernier nécessite l'affiche dans une liste de sélection des différents types disponibles d'où la boucle `foreach($types as $cle=>$type)`.
- proposer des liens permettant d'accéder au détail d'un article (ou au contenu d'un album)
- proposer si nécessaire des liens pour afficher la page précédente et/ou la page suivante. La décision se fera en fonction des valeurs de `$tri->premierArticle()` et `$nbTotalArticles`
- proposer si nécessaire un lien pour revenir à l'album père (si nous ne sommes pas à l'album racine)
- proposer un lien pour ajouter un article

Listing 10.25 : SuperTheque_VueVisualisation_inc.php

```
<form action="" method="post">
<table width="100%">
  <tr>
    <th>
      <a href="?mode=tri&tri_champ=titre&tri_sens=-1">-</a>
      Titre
      <a href="?mode=tri&tri_champ=titre&tri_sens=1">+</a>
    </th>
    <th>
      <a href="?mode=tri&tri_champ=typeId&tri_sens=-1">-</a>
      Type
      <a href="?mode=tri&tri_champ=typeId&tri_sens=1">+</a>
    </th>
  </tr>
  <tr>
    <td>
      <input type="hidden" name="mode" value="filtre" />
      <input type="text" name="filtre_titre" />
    </td>
    <td>
      <select name="filtre_typeId">
        <option value="-1">Tous</option>
        <?php
          foreach($types as $typeId=>$type)
          {
?>
        <option value="<?php echo $typeId;?>"><?php echo $type;?></option>
        <?php
          }
?>
    </select>
```

```
    </td>
    <td>
      <input type="submit" value="Filtrer" />
    </td>
  </tr>
  <tr>
    <td colspan="3"><hr /></td>
  </tr>
  <?php
     if (count($articles)>0) foreach($articles as $articleListe)
     {
 ?>
  <tr>
    <td>
      <?php echo htmlEntities($articleListe->getTitre()); ?>
    </td>
    <td>
      <?php echo $types[$articleListe->getTypeId()]; ?>
    </td>
    <td>
      <?php
          echo "<a
          ✂ href=\"?mode=article&articleId=".$articleListe->getId()."\">";
 ?>
        [D&eacute;tails]
      </a>
    </td>
  </tr>
  <?php
     } // Fin du foreach
 ?>
</table>
</form>
<hr />
<table width="100%">
  <tr>
    <td width="33%" align="left">
      <?php
      if ($plage->getPremierArticle() > 0) {
 ?>
      <a href="?mode=precedents">&lt;&lt;&lt; Articles
      ✂ pr&eacute;c&eacute;dents</a>
      <?php
      }
 ?>
    </td>
    <td width="34%" align="center">
      <?php
      if ($filtre->getAlbumId() != 0)
      {
       // Inserer un lien pour remonter a l'album "pere"
 ?>
```

```
          <a href="?mode=article&articleId=<?php echo $article->getAlbumId();
          ✂ ?>">[..]</a>
          <?php
          }
     ?>
        </td>
        <td align="right">
          <?php
          if ($nbTotalArticles > $plage->getPremierArticle() +
          ✂ $plage->getNbArticleMax()) {
     ?>
          <a href="?mode=suivants">Articles suivants &gt;&gt;&gt;</a>
          <?php
          }
     ?>
        </td>
      </tr>
    </table>
    <br />
    <center>
          <a href="?mode=ajout">
            [Ajouter un article &agrave; cet album]
          </a>
    </center>
```

Comme vous pouvez le constater, il y a très peu de code PHP. Comme quoi, nous avons bien réussi à distinguer la partie traitement de la partie affichage.

Vue ajout d'un nouvel article

Il s'agit du fichier de vue le plus complexe. En effet, cette vue ne doit pas seulement proposer un formulaire de saisie puisqu'elle doit également permettre d'afficher d'éventuels messages d'erreurs détectées lorsque ce formulaire est soumis et dans ce cas les valeurs précédemment saisies doivent être affichées de nouveau.

Listing 10.26 : SuperTheque_VueAjout_inc.php

```
<?php
if (count($erreurs)>0) {
    echo "<ul>";
    foreach($erreurs as $erreur)
    {
        echo "<li>$erreur</li>";
    }
    echo "</ul>";
}
?>
<form action="index.php" method="post">
  <input type="hidden" name="mode" value="sauver" />
  <input type="hidden" name="albumId" value="<?php echo $filtre->getAlbumId();
  ✂ ?>" />
```

```
  <table>
    <tr>
      <td>Titre:</td>
      <td><input type="text" name="titre" value="<?php
echo htmlEntities(stripSlashes($_POST["titre"])); ?>"/></td>
    </tr>
    <tr>
      <td>Type:</td>
      <td>
        <select name="typeId">
          <?php
            foreach($types as $typeId=>$type)
            {
 ?>
          <option value="<?php echo $typeId;?>" <?php if
          ✂ ($_POST["typeId"]==$typeId) echo "selected=\"selected\"";?>><?php
          ✂ echo $type;?></option>
          <?php
            }
 ?>
        </select>
      </td>
    </tr>
    <tr>
      <td>Commentaire:</td>
      <td>
        <textarea name="commentaire" cols="40" rows="5"><?php
echo htmlEntities(stripSlashes($_POST["commentaire"])); ?></textarea>
      </td>
    </tr>
    <tr>
      <td colspan="2">
        <i>Vous pouvez &agrave; votre guise ajouter d'autres informations</i>
      </td>
    </tr>
    <tr>
      <td colspan="2">
        <input type="submit" value="Sauver" />
      </td>
    </tr>
  </table>
</form>
```

Dans le cas d'une saisie invalide, il est important de restituer le formulaire à l'utilisateur dans l'état où il l'a laissé avant de valider sa saisie. Pour cela, nous devons préciser les valeurs par défaut (valeurs précédemment saisies) de chaque champ. Dans le cas d'un champ texte, il suffit de préciser l'attribut `value`. Ainsi nous ne nous contentons pas d'un simple `<input type="text" name="titre" />` mais nous préciserons `<input type="text" name="titre" value="<?php echo htmlEntities(stripSlashes ($_POST["titre" ])); ?>"/>` (la valeur ayant été soumise via la méthode post)

Ceci amène d'ailleurs plusieurs remarques.

Nous retrouvons la problématique du "magic quotes" qui implique l'utilisation de `stripSlashes()`.

htmlEntities

Attention ! Lorsque vous spécifiez un attribut `value`, n'oubliez pas, d'une part, de mettre la valeur entre guillemets (mais ça, vous devriez déjà en avoir l'habitude, sans quoi il est grand temps de s'y mettre !) et, d'autre part, de faire appel à `htmlEntities()` afin, en particulier, de remplacer les guillemets par leurs équivalents HTML pour éviter tout conflit avec les guillemets délimiteurs de la chaîne.

Rappel des précédentes valeurs

Dans le cas d'une liste de sélection, il faut préciser `selected` pour les champs sélectionnés (ce qui en XHTML se déclare par `selected="selected"`) ; dans le cas des boutons radio, il faut préciser `checked` (ce qui en XHTML se déclare par `checked="checked"`) ; et, enfin, pour les zones de texte, il faut préciser leur contenu entre les balises `<textearea>` et `</textarea>`.

Vue détail d'un article

Listing 10.27 : SuperTheque_VueDetail_inc.php

```
<table>
    <tr>
      <td width="10%" >Titre:</td>
      <td><b><?php echo htmlEntities($article->getTitre());?></b></td>
    </tr>
    <tr>
      <td>Type:</td>
      <td><?php echo $types[$article->getTypeId()];?></td>
    </tr>
    <tr>
      <td>Commentaire:</td>
      <td><?php echo nl2br(htmlEntities($article->getCommentaire()));?></td>
    </tr>
    <tr>
      <td colspan="2">
         <i>Vous pouvez &agrave; votre guise compl&eacute;ter cette page pour
         ✂ afficher d'autres informations (que vous aurez pris soin d'ajouter
         ✂ apr&egrave;s avoir compl&egrave;t&eacute; la page de saisie).</i>
      </td>
    </tr>
</table>
<br />
<center>
<a href="?mode=article&articleId=<?php echo $article->getAlbumId();?>">Retour
✂ &agrave; la liste.</a>
</center>
```

Il n'y a pas de réelles nouveautés dans ce script, seule la ligne permettant l'affichage du commentaire peut nécessiter quelques explications.

Retours à la ligne

La zone de texte, définie dans le formulaire pour saisir un commentaire, nous permet de saisir un texte formaté avec des retours à la ligne. Mais il faut garder à l'esprit que ceux-ci sont matérialisés par des '`\n`'. Ainsi, si une description contenant des retours à la ligne est stockée en base de données puis restituée telle quelle (`echo $commentaire`) dans un document HTML, elle perdra sa mise en page. En effet, le caractère '`\n`' n'est pas considéré en HTML comme un retour à la ligne, la balise HTML correspondante étant `<br>` (ou `<br/>` en XHTML). Il faut donc utiliser, avant affichage, la fonction `nl2br()`. C'est donc ce que nous utilisons ici.

10.7. Access (MS)

Bien que l'on ne puisse pas considérer MS Access comme un véritable serveur de bases de données, c'est certainement la base de données la plus connue du grand public.

Installation

Afin de pouvoir accéder à une base de données Access depuis PHP, vous devez impérativement établir une liaison ODBC.

L'opération s'avère extrêmement simple :

1. Ouvrez le Panneau de configuration.

Figure 10.4 : *Panneau de configuration*

2. Sélectionnez *Performances et maintenance.*

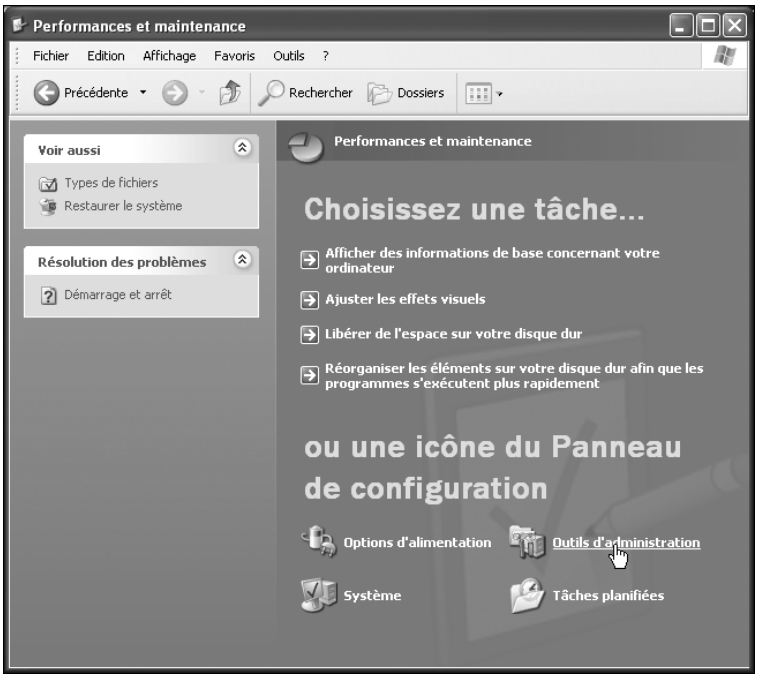


Figure 10.5 : *Performances et maintenance*

3. Sélectionnez *Outils d'administration*.

Figure 10.6 : *Outils d'administration*

4. Voilà, vous êtes en passe de découvrir le panneau d'administration de "sources de données (ODBC)". Cliquez...

Figure 10.7 : *Administrateur de sources de données ODBC*

5. Sélectionnez l'onglet **Sources de données système** puis, sans faiblir, cliquez sur *Ajouter*.

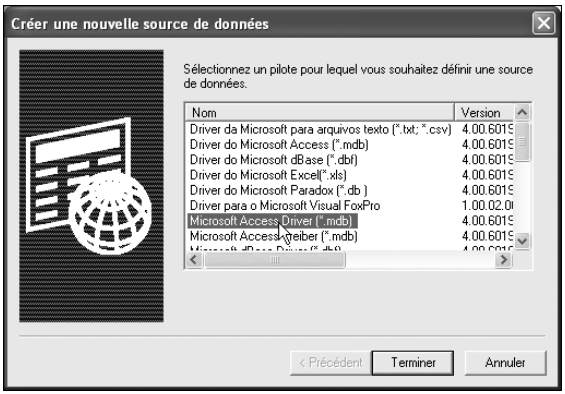


Figure 10.8 : *Créer une nouvelle source de données*

6. Nous voilà au cœur du sujet. Sélectionnez le pilote correspondant à votre type de base de données, en l'occurrence Microsoft Access Driver, puis cliquez sur *Terminer*.

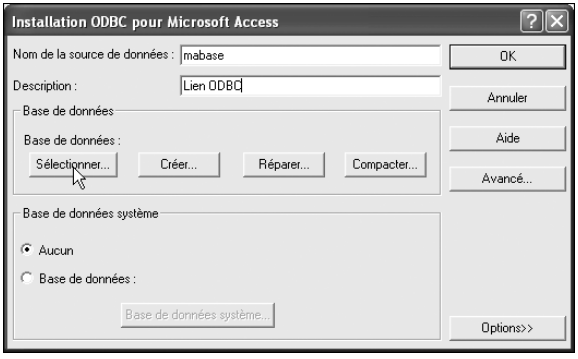


Figure 10.9 : *Installation ODBC pour Microsoft Access*

7. Vous pouvez maintenant saisir un nom pour ce lien ODBC (qui peut être différent du nom du fichier Access) qui servira de référence pour les futures connexions via PHP ainsi qu'un commentaire. Cliquez ensuite sur le bouton **Sélectionner....**

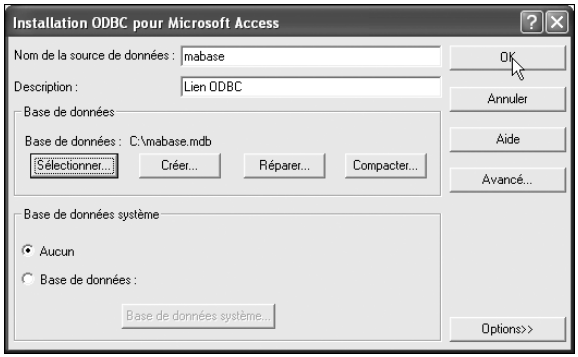


Figure 10.10 : *Installation ODBC pour Microsoft Access (suite)*

8. Vous êtes alors invité à repérer le fichier de votre base de données dans l'arborescence de votre disque dur. Une fois que cela est fait, c'est bon, vous n'avez plus qu'à cliquer sur OK.

Figure 10.11 : *Voilà, c'est fait*

9. Et voilà, l'onglet **Sources de données système** s'est enrichi d'une nouvelle ligne.

Utilisation

Évidemment, les fonctions permettant d'accéder à MS Access sont celles proposées pour les liaisons ODBC.

Vous êtes donc invité à vous reporter à la section ODBC.

10.8. DB2 (IBM)

DB2 est la base de données professionnelle du géant de l'informatique, IBM.

Installation

Le but de ce chapitre n'est pas de faire de vous un administrateur de base de données DB2. Non. Nous nous contenterons de décrire une installation standard, sans tenir compte des problèmes d'optimisation et de sécurité. Le but étant que vous puissiez installer PHP et DB2 sur des machines de test pour vous familiariser avec cet environnement avant de passer à un serveur de bases de données destiné à la production.

Pré-requis

Pour commencer, vous devez vous procurer le script d'installation. Celui-ci est disponible sur le site d'IBM (**http://www.ibm.fr**) à l'adresse **http://www-5.ibm.com/fr/software/data/db2/index.html**.

Pour la version Linux, il s'agit d'un fichier *db2pelnx.tar*.

Installation du serveur de bases de données

Nous supposerons ici que le serveur de bases de données et le serveur web tournent sur la même machine.

Sous Linux

La première chose à faire consiste à décompresser l'archive dans un répertoire quelconque (ex. : */tmp/db2*),

```
# tar xvf db2pelnx.tar
```

puis à lancer le script d'installation :

```
# ./db2setup
```

Apparaît alors le premier écran, proposant une liste d'éléments à installer :

```
+------------------------------ Install DB2 V7 ----------------------------+
|                                                                          |
|   Select the products you are licensed to install.  Your Proof of        |
|   Entitlement and License Information booklet identify the products for  |
|   which you are licensed.                                                |
|                                                                          |
|   To see the preselected components or customize the selection, select   |
|   Customize for the product.                                             |
|   [*] DB2 Administration Client : Customize... :                         |
|   [*] DB2 UDB Personal Edition : Customize... :                          |
|   [*] DB2 Application Development Client : Customize... :                |
|                                                                          |
|   To choose a language for the following components, select Customize for  |
```

```
|   the product.                                                              |
|       DB2 Product Messages                          [ Customize... ]        |
|       DB2 Product Library                           [ Customize... ]        |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|   [   OK   ]                  [ Cancel ]                    [  Help  ]      |
+-----------------------------------------------------------------------------+
```

1. Sélectionnez les trois modules, puis sélectionner l'option "`customize`" de "`DB2 Application Development Client`" et assurez vous de bien cocher l'option "*Create links for DB2 libraries*".
2. Lancez l'installation en sélectionnant le bouton OK.
3. Deux écrans successifs apparaissent alors, vous invitant à préciser les noms et paramètres des différents comptes DB2. Conservez les paramètres par défaut et validez.

   Vous aurez alors, trois nouveaux comptes :

   - `db2inst1` associé au groupe `db2iadm1`, avec pour répertoire */home/db2inst1* et mot de passe `ibmdb2` ;
   - `db2fenc1` associé au groupe `db2fadm1`, avec pour répertoire */home/db2fenc1* et mot de passe `ibmdb2` ;
   - `db2as` associé au groupe `db2asgrp`, avec pour répertoire */home/db2as* et mot de passe `ibmdb2`.

Et voilà, votre base de données a été installée sous */usr/IBMdb2/V7.1*.

Démarrage du serveur de bases de données

À l'issue de l'installation, le serveur de bases de données tourne. Sachez toutefois que, si vous avez à lancer le serveur, cela se fait simplement à partir du compte `db2inst1` avec la commande

```
$ db2start
```

Test du serveur de bases de données

À partir du compte `db2inst1`, vous pouvez aisément tester le bon fonctionnement du serveur de bases de données.

En créant la base de données d'exemple :

```
$ db2sampl
```

En vous y connectant et en testant une requête :

```
$ db2 connect to sample
$ db2 "select * from staff where dept=20"
```

Vous devez alors avoir quatre lignes de résultat. Vous pouvez alors vous déconnecter.

```
$ db2 connect reset
```

Création d'une base

Vous pouvez créer une base de données depuis le client db2.

```
$db2 create database <nom de la base>
```

Arrêt du serveur de bases de données

Vous pouvez arrêter le serveur de bases de données avec l'instruction

```
$ db2stop
```

Installation du client de base de données sur le serveur web

Nous avons, ici, supposé que le serveur de bases de données tournait sur la même machine que le serveur web. Nous pourrions donc nous attendre à ce qu'il n'y ait rien d'autre de particulier à faire.

Malheureusement, nous avons pu constater (du moins dans le cas de figure qui nous intéressait) un léger dysfonctionnement dans la procédure d'installation. En effet, nous nous sommes retrouvés avec un lien symbolique */usr/local/lib/libbd2.so.3* pointant sur un fichier */usr/local/lib/libbd2.so* inexistant. Ce qui nous posait problème lors de la compilation d'Apache. Problème que nous avons simplement résolu en créant un lien symbolique du fichier */usr/db2inst1/lib/libdb2.so* vers */usr/lib*, comme suit :

```
# ln -s /usr/db2inst1/lib/libdb2.so /usr/lib
```

Configuration de PHP avec support DB2

Sous Linux

Pour accéder à une base de données DB2 depuis PHP, vous n'avez qu'à recompiler PHP avec l'option `--with-ibm-db2=/usr/db2inst1/sqllib`.

Vous pouvez vous reporter au chapitre "Installation" pour plus de détails sur la façon de compiler PHP.

Vérification de l'installation

Vous pouvez maintenant tester un script ne contenant que le code `<?php phpinfo(); ?>` et vous devez apercevoir les lignes suivantes :

odbc

ODBC Support	enabled
Active Persistent Links	0
Active Links	0
ODBC library	db2
ODBC_INCLUDE	-I/usr/db2inst1/sqllib/include
ODBC_LFLAGS	-L/usr/db2inst1/sqllib/lib
ODBC_LIBS	-ldb2

Directive	Local Value	Master Value
odbc.allow_persistent	On	On
odbc.check_persistent	On	On
odbc.default_db	*no value*	*no value*
odbc.default_pw	*no value*	*no value*
odbc.default_user	*no value*	*no value*
odbc.defaultbinmode	return as is	return as is
odbc.defaultlrl	return up to 4096 bytes	return up to 4096 bytes
odbc.max_links	Unlimited	Unlimited
odbc.max_persistent	Unlimited	Unlimited

Figure 10.12 : *phpinfo() pour DB2*

Utilisation

Bien que n'utilisant pas une liaison ODBC, les fonctions permettant d'accéder aux bases DB2 sont les mêmes que celles proposées pour les liaisons ODBC.

Vous êtes donc invité à vous reporter à la section ODBC.

10.9. MySQL

MySQL est probablement la base de données la plus connue des adeptes de PHP. Le fait qu'elle soit sous licence GPL, disponible sous différentes plateformes et, somme toute, assez complète et performante n'y est certainement pas pour rien. C'est très probablement d'ailleurs la base de données que votre hébergeur vous propose (sous un environnement Linux, afin d'allier la performance à la limitation des frais de licence et de maintenance).

MySQL 3.X
La dernière version de production de MySQL est la version 4.0. C'est donc l'installation de cette dernière qui est décrite ici. Toutefois, il est possible que votre


REMARQUE

hébergeur propose toujours la version 3 et donc que vous souhaitiez installer la même version. Rassurez-vous, la procédure d'installation est totalement identique.

Installation

Nous nous contenterons ici de décrire une installation standard, sans prendre en compte les problèmes d'optimisation et de sécurité. Le but étant que vous puissiez installer PHP et MySQL sur des machines de test pour vous familiariser avec cet environnement avant de passer à un serveur de bases de données destiné à la production.

Pré-requis

Si vous êtes sous Windows et avez installé EasyPHP, vous n'aurez rien à installer ou configurer, puisque tout est fait automatiquement (aussi bien pour ce qui concerne MySQL que pour ce qui concerne PHP). Avec EasyPHP 1.7, c'est MySQL 4.0.15 qui est fourni.

Dans les autres cas, pour commencer, vous devrez vous procurer l'archive de MySQL, disponible sur le site officiel à l'adresse **http://www.mysql.com/downloads/** (vous la trouverez également sur le CD-ROM fourni).

Installation du serveur de bases de données

Sous Windows

Vous devriez maintenant avoir un paquetage avec un nom du genre : `mysql-4.0.14b-win.zip`.


INTERNET

Winzip
Si vous n'avez pas Windows XP ou un utilitaire pour extraire les fichiers du paquetage, vous pouvez télécharger la version de démonstration de Winzip sur ***http://www.winzip.com****.*

Il faut ensuite extraire les fichiers du paquetage dans un répertoire quelconque et exécuter le fichier *Setup.exe*.

L'installation est très simple et rapide :

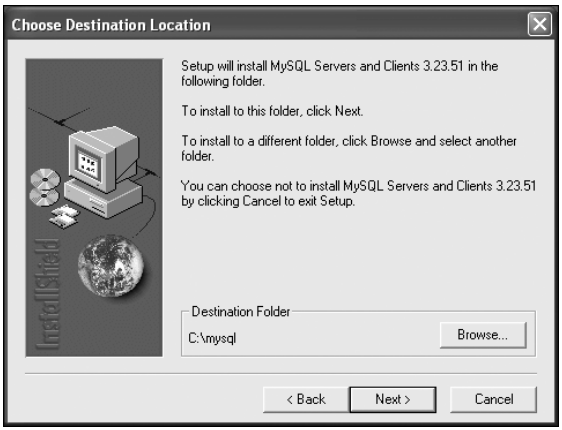


Figure 10.13 : *Répertoire d'installation*

Nous choisirons une installation typique, qui nous conviendra pour ce que l'on souhaite faire.

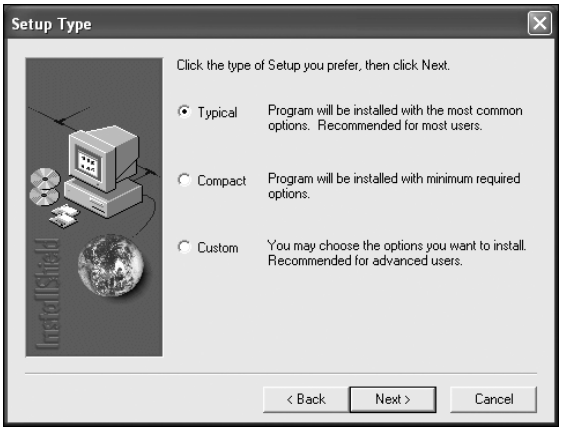


Figure 10.14 : *Installation typique*

L'installation crée automatiquement un fichier *my.ini* dans le répertoire de Windows, qui servira pour utiliser MySQL en tant que service.

Installation

Si vous installez MySQL dans un répertoire autre que C:\MYSQL, ou si vous voulez installer MySQL en tant que service sur Windows 2000/NT/XP, il faudra créer un fichier appelé C:\MY.CNF ou placer un fichier my.ini dans votre répertoire Windows avec les lignes suivantes (à modifier selon votre configuration) :

```
[mysqld]
basedir=C:/repertoireinstallation/
datadir=C:/repertoiredonnees/
```

Quoi qu'il en soit, une fois l'installation de MySQL terminée, le répertoire d'installation contiendra des fichiers appelés my-xxxxx.cnf qu'il est possible d'utiliser


REMARQUE

pour refaire votre propre fichier C:\MY.CNF, les différents fichiers dépendent de la taille de votre base de données : de small pour les petites bases de données à huge pour les très grosses en passant par medium et large.

Sous Linux

Il existe plusieurs façons de faire. Mais les deux principales consistent soit à utiliser les versions précompilées de MySQL soit à les compiler soit même. Mais curieusement l'arborescence des fichiers obtenus différent légèrement entre les deux solutions.

Version précompilée

Une fois l'archive correspondant à votre environnement (ici, une architecture intel) installée sur votre disque dur (par exemple sous /usr/local), il vous faudra la décompresser par la commande :

```
# tar zxvf mysql-standard-4.0.20-pc-linux-i686.tar.gz
```

Vous disposez alors de tous les fichiers MySQL sous */usr/local/mysql-standard-4.0.20-pc-linux-i686*.répertoire que nous renommerons en */usr/local/mysql*.

```
# cd /usr/local
# mv mysql-standard-4.0.20-pc-linux-i686 mysql
# cd mysql
```

Il vous faut, désormais, initialiser l'espace destiné à accueillir les bases de données.

```
# ./scripts/mysql_install_db
```

Assurez-vous de bien exécuter le script depuis le répertoire mysql. Cela se traduit par la création de fichiers et de répertoires sous */usr/local/mysql/data*.

Puisqu'il est préférable de ne pas faire tourner le serveur MySQL sous le compte `root`, nous vous invitons maintenant à créer un compte `mysql`.

```
# groupadd mysql
# useradd -g mysql -s /bin/bash mysql
```

Vous pouvez maintenant modifier le propriétaire des fichiers de base de données :

```
# chown -R mysql /usr/local/mysql/data
```

ainsi que le groupe auquel appartiennent les fichiers du serveur.

```
# chgrp -R mysql /usr/local/mysql
```

Copiez ensuite le fichier de configuration dans le repertoire */etc/*.

```
# cp support-files/my-medium.cnf /etc/my.cnf
```


REMARQUE

Compte sans mot de passe
Dans l'état actuel, ce compte `mysql` *ne possède pas de mot de passe ; vous ne pourrez y accéder que depuis le compte* `root` *en tapant* `su - mysql`.

Version à compiler

Pour cela vous devez récupérer les sources du serveur MySQL (disponibles sur le CD-ROM) puis, une fois l'archive installée sur votre disque dur, il vous faudra la décompresser par la commande :

```
# tar zxvf mysql-4.0.20.tar.gz
```

Une fois les sources décompressées, il est possible de les compiler par les commandes :

```
# cd mysql-4.0.20
# ./configure --prefix=/usr/local/mysql
# make
# make install
```

MySQL est a présent installé sous */usr/local/mysql*.

```
# cd /usr/local/mysql
```

Il vous faut, désormais, initialiser l'espace destiné à accueillir les bases de données.

```
# ./bin/mysql_install_db
```

Cela se traduit par la création de fichiers et de répertoires sous */usr/local/mysql/var*.

Puisqu'il est préférable de ne pas faire tourner le serveur MySQL sous le compte `root`, nous vous invitons donc à créer un compte `mysql`.

```
# groupadd mysql
# useradd -g mysql -s /bin/bash mysql
```

Vous pouvez maintenant modifier le propriétaire des fichiers de base de données :

```
# chown -R mysql /usr/local/mysql/var
```

ainsi que le groupe auquel appartiennent les fichiers du serveur.

```
# chgrp -R mysql /usr/local/mysql
```

Copiez ensuite le fichier de configuration dans le répertoire */etc/*.

```
# cp share/mysql/my-medium.cnf /etc/my.cnf
```


REMARQUE

Compte sans mot de passe
Dans l'état actuel, ce compte `mysql` *ne possède pas de mot de passe ; vous ne pourrez y accéder que depuis le compte* `root` *en tapant* `su - mysql`.

Démarrage du serveur de bases de données

Sous Windows

Pour lancer le service, il faut exécuter le fichier *winmysqladmin.exe* du répertoire *bin* de MySQL. À la première exécution, le programme vous demandera un nom d'utilisateur et un mot de passe à créer. Un feu vert devrait alors apparaître dans votre barre des tâches, indiquant que le serveur MySQL est prêt.

Sous Linux

Le démon `mysqld` doit être actif pour pouvoir utiliser la base de données MySQL.

```
# /usr/local/mysql/bin/mysqld_safe --user=mysql &
```

Avec MySQL 3.X vous devrez lancer la commande

```
# /usr/local/mysql/bin/safe_mysqld --user=mysql &
```

Création d'une base

Sous Windows

En utilisant le logiciel WinMYSQLAdmin et l'onglet **Databases**, il est possible de voir les bases de données existantes. Pour en créer une, il suffit de cliquer avec le bouton droit sur le nom de la machine, puis de sélectionner *Create Database*. Une fenêtre s'ouvre, demandant le nom de la nouvelle base de données.

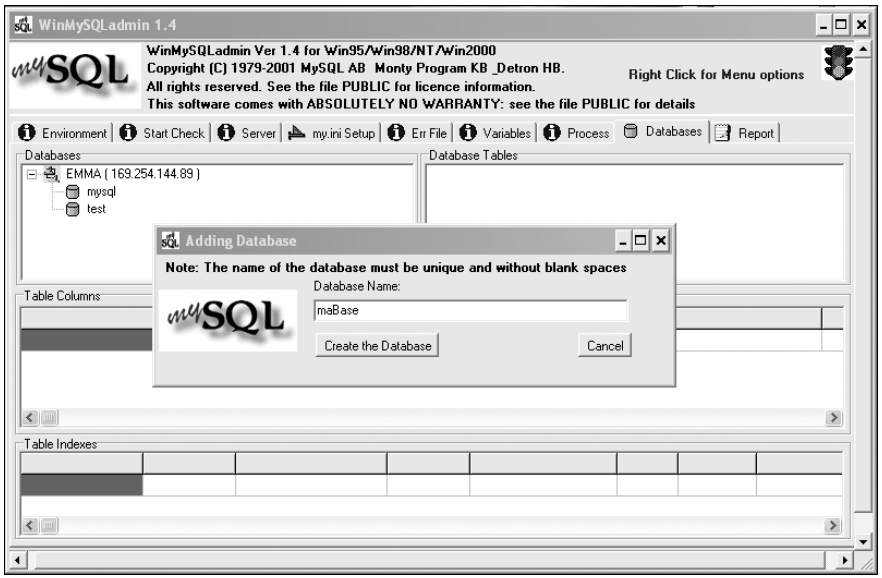


Figure 10.15 : *Ajouter une base de données*

Une autre façon de faire consiste à exécuter la commande en ligne `mysql` puis de taper :

```
CREATE DATABASE maBase;
```

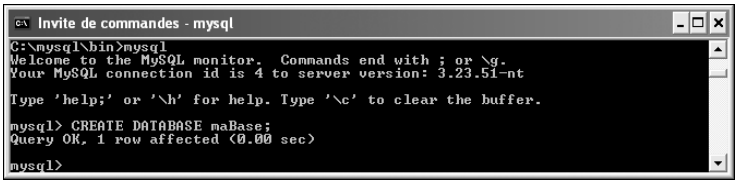


Figure 10.16 : *Ajouter une base de données par la commande en ligne*

Sous Linux

Une fois `mysql` lancé, il suffit d'entrer la commande :

```
$ mysql
CREATE DATABASE maBase;
```

Utiliser un autre utilisateur que l'utilisateur courant
Pour lancer `mysql` avec un autre nom d'utilisateur que le nom courant, il faut passer l'option `-u`. Il est également possible de passer le mot de passe en ligne de commande par l'option `-p`.

```
$ mysql –u root –p motdepasse
```

Test du serveur de bases de données

Il est possible de faire exécuter quelques requêtes SQL pour voir le fonctionnement du serveur. Il suffit de lancer `mysql` et de taper, par exemple, (cela suppose que vous ayez créé la base de données `maBase` auparavant) :

```
$ mysql
USE maBase;
CREATE Table maTable (id INTEGER, valeur VARCHAR(32));
INSERT INTO maTable (1, "toto");
INSERT INTO maTable (2, "titi");
SELECT * FROM maTable;
```

Voici une capture d'écran sous Windows de cette série de requêtes :

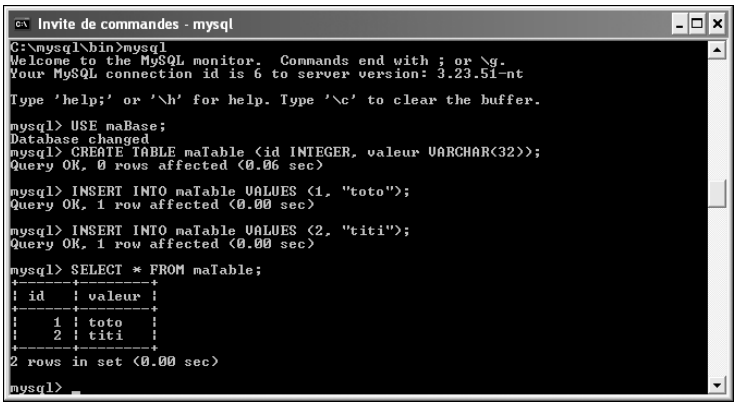


Figure 10.17 : *Quelques requêtes pour tester*

Configuration par défaut de MySQL

Avec une installation par défaut de MySQL, vous aurez tous les droits utilisateurs si la base de données tourne localement, et que vous utilisez pour nom d'utilisateur "`root`" et pour mot de passe simplement une chaîne vide. Mais n'hésitez surtout pas à modifier cette configuration de base pour obliger l'utilisation d'un mot de passe. Cette opération peut se réaliser en exécutant la requête SQL suivante sur la base `mysql` :

```
UPDATE user SET password=PASSWORD('<mot de passe>') WHERE user=root
```

Configuration de PHP avec support MySQL

Les dernières versions de PHP intégraient par défaut le support de MySQL dans le noyau. Ce n'est plus vrai depuis PHP 5. Et bien qu'il n'y avait, a priori, rien à faire pour que cela fonctionne (hormis avoir un serveur MySQL à disposition) avec les versions 4.2 et 4.3, il était déjà conseillé de recompiler PHP avec l'API de la version MySQL installée. Avec PHP 5 cela devient une nécessité. Pour cela, il suffit d'ajouter l'option `--with-mysql=/usr/local/mysql`.

Vous pouvez vous reporter au chapitre "Installation" pour plus de détails sur la compilation de PHP.

Configuration php.ini

Le fichier *php.ini* permet de définir les paramètres suivants

```
mysql.allow_persistent = On
```

Autorise ou interdit les connexions persistantes.

```
mysql.max_persistent = -1
```

Nombre maximum de connexions persistantes. -1 autorise un nombre illimité.

```
mysql.max_links = -1
```

Nombre maximum de connexions (persistantes ou non). -1 autorise un nombre illimité.

```
mysql.default_port =
```

Indique le port à utiliser s'il n'est pas précisé lors de l'appel à `mysql_connect()` ou à `mysql_pconnect()`. Si vous ne renseignez pas cette directive, `mysql_connect` utilisera le port indiqué par `$MYSQL_TCP_PORT`, sinon c'est le port associé au service `mysql-tcp`, indiqué dans */etc/services*, qui sera utilisé, ou, enfin, la valeur définie lors de la compilation par la constante `MYSQL_PORT`. Sous Windows, seul le port précisé par `MYSQL_PORT` est pris en compte.

```
mysql.default_socket =
```

Indique le nom de la socket par défaut pour les connexions MySQL en local. Si vous ne renseignez pas cette directive, PHP utilisera la configuration MySQL par défaut.

```
mysql.default_host =
```

Indique le nom (ou l'adresse) du serveur mysql à utiliser par défaut s'il n'est pas précisé lors de l'appel à `mysql_connect()` ou `mysql_pconnect()`. Cette option n'est pas prise en compte en mode sécurisé.

```
mysql.default_user =
```

Indique le nom d'utilisateur à donner par défaut s'il n'est pas précisé lors de l'appel à `mysql_connect()` ou `mysql_pconnect()`.Cette option n'est pas prise en compte en mode sécurisé.

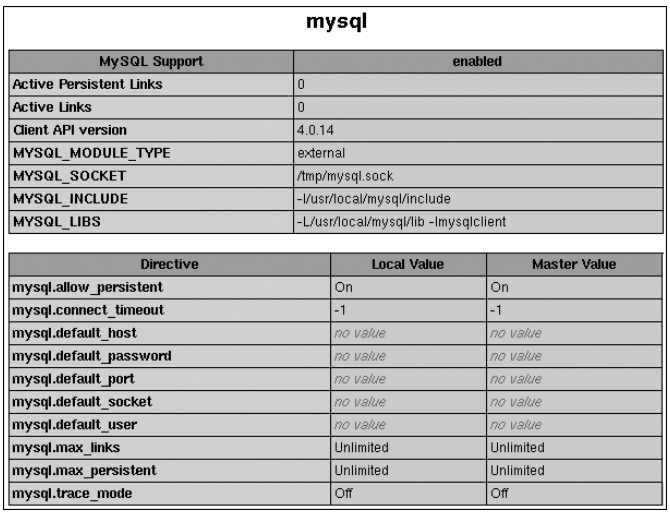
```
mysql.default_password =
```

Indique le mot de passe à utiliser par défaut s'il n'est pas précisé lors de l'appel à `mysql_connect()` ou à `mysql_pconnect()`. Cette option n'est pas prise en compte en mode sécurisé.

Il est déconseillé de stocker les mots de passe dans ce fichier. N'importe quel utilisateur ayant accès à votre PHP, ou ayant les droits de lecture, pourrait lancer la commande `echo cfg_get_var("mysql.default_password")`, et accéder ainsi à ce mot de passe.

Vérification

Vous pouvez maintenant tester un script ne contenant que le code `<?php phpinfo(); ?>`. Vous devez apercevoir les lignes suivantes :



mysql

MySQL Support	enabled
Active Persistent Links	0
Active Links	0
Client API version	4.0.14
MYSQL_MODULE_TYPE	external
MYSQL_SOCKET	/tmp/mysql.sock
MYSQL_INCLUDE	-I/usr/local/mysql/include
MYSQL_LIBS	-L/usr/local/mysql/lib -lmysqlclient

Directive	Local Value	Master Value
mysql.allow_persistent	On	On
mysql.connect_timeout	-1	-1
mysql.default_host	*no value*	*no value*
mysql.default_password	*no value*	*no value*
mysql.default_port	*no value*	*no value*
mysql.default_socket	*no value*	*no value*
mysql.default_user	*no value*	*no value*
mysql.max_links	Unlimited	Unlimited
mysql.max_persistent	Unlimited	Unlimited
mysql.trace_mode	Off	Off

Figure 10.18 : *phpinfo()*

Utilisation

Comme indiqué dans le chapitre d'introduction, l'utilisation basique de la base de données s'opère selon le schéma suivant :

- Connexion grâce aux fonctions `mysql_connect()` ou `mysql_pconnect()` et `mysql_select_db()`.
- Soumission de la requête via la fonction `mysql_query()`.
- Déconnexion grâce à la fonction `mysql_close()` (dans le cas d'une connexion non persistante).

Connexion

La connexion à une base de données MySQL nécessite deux opérations. Il faut, dans un premier temps, se connecter au serveur de bases de données avec la fonction `mysql_connect()`; la sélection de la base s'opère dans un second temps avec `mysql_select_db()`.

mysql_connect()

Établit une connexion (non persistante) avec le serveur de bases de données.

Syntaxe

```
resource mysql_connect([string $nomServeur [:$port]
[:$cheminSocket] [, string $utilisateur [, string $motDePasse
[, boolean $nouvelleCnx [, int $option]]]]])
```

`$nomServeur` Nom du serveur (par défaut "localhost").

`$port`	Port sur lequel la connexion au serveur de bases de données s'effectue (par défaut 3306).
`$cheminSocket`	Chemin vers le fichier *socket* (lorsque le serveur de bases de données tourne en local).
`$utilisateur`	Nom d'utilisateur (sous Linux/UNIX ; par défaut, le propriétaire du serveur web, qui traditionnellement pour Apache est "nobody").
`$motDePasse`	Mot de passe de l'utilisateur (par défaut, aucun mot de passe).
`$nouvelleCnx`	Précise si l'on souhaite (`TRUE`) ou non (`FALSE`, valeur par défaut) forcer la création d'une nouvelle connexion si une connexion similaire a déjà été ouverte.
`$option`	Précise une option de communication avec le serveur de base de données. Ce paramètre est disponible depuis la version 4.3.0 de PHP. Elle peut prendre l'une des valeurs suivantes: `MYSQL_CLIENT_COMPRESS`, pour envoyer les données sous forme compressées. `MYSQL_CLIENT_IGNORE_SPACE`, pour autoriser les espaces après les noms de fonction (tous les noms de fonctions sont alors des mots réservés). `MYSQL_CLIENT_INTERACTIVE`, pour baser le time-out de déconnexion non pas sur la valeur de `wait_timeout` mais sur celle de `interactive_timeout`. `MYSQL_CLIENT_SSL`, (uniquement avec MySQL 4) pour utiliser le protocole SSL.
`retour`	Identifiant de connexion au serveur de bases de données, ou `FALSE` en cas d'échec.

L'utilisation d'une connexion persistante requiert, à la place, l'appel à la fonction `mysql_pconnect()`.

mysql_pconnect()

Établit une connexion persistante avec le serveur de bases de données. Si une connexion persistante est disponible, elle sera utilisée ; sinon, une nouvelle connexion persistante sera créée.

Syntaxe	`resource mysql_pconnect([string $nomServeur [:$port] [:$cheminSocket] [, string $utilisateur [, string $motDePasse [, int $option]]])`
`$nomServeur`	Nom du serveur (par défaut "localhost").
`$port`	Port sur lequel la connexion au serveur de bases de données s'effectue (par défaut 3306).
`$cheminSocket`	Chemin vers le fichier *socket* (lorsque le serveur de bases de données tourne en local).
`$utilisateur`	Nom d'utilisateur (sous Linux/UNIX ; par défaut, le propriétaire du serveur web, qui traditionnellement pour Apache est "nobody").
`$motDePasse`	Mot de passe de l'utilisateur (par défaut, aucun mot de passe).

`$option`	Précise une option de communication avec le serveur de base de données. Ce paramètre est disponible depuis la version 4.3.0 de PHP. Elle peut prendre l'une des valeurs suivantes: `MYSQL_CLIENT_COMPRESS`, pour envoyer les données sous forme compressées. `MYSQL_CLIENT_IGNORE_SPACE`, pour autoriser les espaces après les noms de fonction (tous les noms de fonctions sont alors des mots réservés). `MYSQL_CLIENT_INTERACTIVE`, pour baser le time-out de déconnexion non pas sur la valeur de `wait_timeout` mais sur celle de `interactive_timeout`. `MYSQL_CLIENT_SSL`, (uniquement avec MySQL 4) pour utiliser le protocole SSL.
`retour`	Identifiant de connexion au serveur de bases de données, ou `FALSE` en cas d'échec.

mysql_select_db

Sélectionne une base de données.

Syntaxe	`boolean mysql_select_db(string $baseDonnees [, resource $idConnexion])`
`$baseDonnees`	Nom de la base de données.
`$idConnexion`	Identifiant de connexion au serveur de bases de données tel que retourné par `mysql_connect()`. Par défaut, c'est l'identifiant de la dernière connexion qui est utilisé.
`retour`	`TRUE` en cas de succès, `FALSE` sinon (ex. : base de données inexistante).

Exécution de la requête SQL

Pour exécuter une requête SQL, il suffit de faire appel à la fonction `mysql_query()`.

mysql_query()

Exécute une requête SQL.

Syntaxe	`resource mysql_query(string $requete [, resource $idConnexion])`
`$requete`	Requête SQL.
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `mysql_connect()` ou `mysql_pconnect()`. Par défaut, c'est l'identifiant de la dernière connexion qui est utilisé.
`retour`	Identifiant de requête SQL, ou `FALSE` en cas d'échec.

Dans le cas d'une requête retournant un résultat (typiquement une requête `SELECT`), il faudra également analyser le résultat. Mais nous verrons cela un peu plus loin.

Déconnexion

Pour vous déconnecter – opération théoriquement facultative mais toutefois vivement conseillée –, vous disposez de la fonction `mysql_close()`.

mysql_close

Met fin à la connexion à la base de données et libère les ressources associées.

Syntaxe	`boolean mysql_close([resource $idConnexion])`
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `mysql_connect()`. Par défaut, c'est l'identifiant de la dernière connexion qui est utilisé.
`retour`	`TRUE` si l'opération a été effectuée avec succès, `NULL` (mais pas strictement `FALSE`) en cas d'échec.

Premier exemple

Vérifiez les paramètres
Prenez garde ! Avant d'exécuter ce script, assurez-vous que les paramètres prédéfinis ne risquent pas de conduire à la suppression des données d'une de vos bases qui, par coïncidence, existerait déjà.

Comme cela est fortement conseillé, nous avons, ici, isolé les paramètres de connexion dans un fichier aisément repérable,

Listing 10.28 : parametres_bd_inc.php

```
<?php

    // Parametres de connexion à la base
    // de données

    $serveur     = "localhost";
    $port        = "";          // doit commencer par :
    $utilisateur = "root";
    $motDePasse  = "";

    $base        = "maBase";

?>
```

et qui est utilisé par le script principal.

Listing 10.29 : insert01.php

```
<?php

    // Paramètres du script
    require_once("parametres_bd_inc.php");
    $table = "tableexemple";

    // Inclusion du script contenant les fonctions
    require_once("insert01_bd_inc.php");

    // Connexion à la base de données
    $idConnexion = mysql_pconnect($serveur, $utilisateur, $motDePasse);
    if (!$idConnexion) {
        die("<b>Impossible de se connecter à la base de données</b>");
    }
    if (!mysql_select_db($base)) {
        die("<b>Impossible de se connecter à la base de données</b>");
    }

    // Appel de la fonction principale
    if (EX_initialiseBD($idConnexion, $table)) {
        echo "Voilà, une nouvelle table avec quelques données, ".
             "vous pouvez le vérifier avec votre client de base ".
             " de données ou via les scripts suivants.";
    } else {
        echo "La création ou l'alimentation de la table à échouée.";
    }

    // Pas de déconnexion dans le cas d'une connexion persistante
    // mysql_close($idConnexion);
?>
```

Ce script utilise les fonctions définies dans le fichier :

Listing 10.30 : insert01_bd_inc.php

```
<?php

    /**
     * Fonction chargée de créer et d'alimenter une table
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la table
     **/
    function EX_initialiseBD($idConnexion, $table)
    {
        // Supprime la précédente table
```

```
        $requete = "DROP TABLE $table";
        @mysql_query($requete, $idConnexion);

        // Crée la table
        $requete = "CREATE TABLE $table (filmId INTEGER ".
                                        " AUTO_INCREMENT PRIMARY KEY,".
                                    "film VARCHAR(64))";
        if (!mysql_query($requete, $idConnexion)) return FALSE;

        // Ajoute quelques données
        $requete = "INSERT INTO $table (film) VALUES ('Forrest Gump')";
        if (!mysql_query($requete, $idConnexion)) return FALSE;
        $requete = "INSERT INTO $table (film) VALUES ('Matrix')";
        if (!mysql_query($requete, $idConnexion)) return FALSE;
        $requete = "INSERT INTO $table (film) VALUES ('La cité de la peur')";
        if (!mysql_query($requete, $idConnexion)) return FALSE;

        return TRUE;
    }
?>
```

Valeur du champ auto-incrémenté

Parfois, après avoir inséré des données dans une table contenant un champ auto-incrémenté, vous aurez besoin de connaître la valeur qui a été donnée à ce champ. Pour cela, vous devrez appeler la fonction `mysql_insert_id()`.

mysql_insert_id()

Retourne la valeur affectée au champ auto-incrémenté lors de la dernière requête `INSERT`.

Syntaxe	`int mysql_insert_id([resource $idConnexion])`
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `mysql_connect()`. Par défaut, c'est l'identifiant de la dernière connexion qui est utilisé.
`retour`	La valeur du champ.

Lecture des enregistrements

Dans le cas d'une requête retournant une liste de données, il convient de récupérer l'identifiant de requête et de l'utiliser pour lire, en boucle, ligne après ligne, chaque enregistrement. Il existe diverses fonctions, chacune permettant de récupérer les champs des enregistrements sous différentes formes.

Les informations peuvent ainsi être retournées :

- Champ par champ ;
- Enregistrement par enregistrement (sous la forme d'un tableau indexé, d'un tableau associatif ou encore d'un tableau à la fois indexé et associatif).

Lecture champ par champ

Bien que cela présente peu d'intérêt, sachez qu'il existe une fonction appelée `mysql_result()` permettant d'accéder au résultat champ par champ.

Lecture enregistrement par enregistrement

La forme la plus simple est celle retournée par la fonction `mysql_fetch_row()`.

mysql_fetch_row()

Retourne un enregistrement, retourné par une requête SQL, sous la forme d'un tableau indexé.

Syntaxe	`array mysql_fetch_row(resource $idResultat)`
`$idResultat`	Identifiant de requête tel que retourné par `mysql_query()`.
`retour`	Tableau indexé contenant autant d'éléments que de champs retournés par la requête ; `FALSE` s'il n'y a plus d'enregistrement à lire, ou rien en cas d'identifiant de requête non valide.

Listing 10.31 : select_eei_bd_inc.php

```
<?php

    /**
     * Fonction listant le contenu d'une table
     * contenant 2 champs (filmId et film)
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la table
     **/
    function EX_listeContenu($idConnexion, $table)
    {
        // Requete
        $requete = "SELECT * FROM $table";
        $idResultat = mysql_query($requete, $idConnexion);
        if (!$idResultat) return FALSE;

        // Boucle de lecture (et d'affichage) des enregistrements
        while ($enreg = mysql_fetch_row($idResultat)) {
            echo "FilmId=".$enreg[0]." ".
                 "Film =" .$enreg[1]."<br />";
```

```
        }
        return TRUE;
    }
?>
```

Mais il est également possible de retourner un enregistrement sous la forme d'un tableau indexé et associatif avec, pour clés, les noms des champs.

mysql_fetch_array()

Retourne un enregistrement, retourné par une requête SQL, sous la forme d'un tableau indexé et/ou associatif.

Syntaxe	`array mysql_fetch_array(resource $idResultat [, int $mode])`
`$idResultat`	Identifiant de requête tel que retourné par `mysql_query()`.
`$mode`	Au choix l'une des constantes suivantes : `MYSQL_ASSOC`, pour retourner un tableau associatif, où les clés sont les noms des champs (faites attention à la casse). `MYSQL_NUM`, pour retourner un tableau indexé. `MYSQL_BOTH` (par défaut), pour retourner un tableau à la fois indexé et associatif, où les clés (en minuscules) sont les noms des champs.
`retour`	Tableau indexé et/ou associatif selon le mode choisi ; `FALSE` s'il n'y a plus d'enregistrement ou rien en cas d'erreur (si l'identifiant de requête n'est pas valide).

C'est ce mode de lecture des données que nous privilégierons ; il est en effet fort pratique pour les champs dont on connaît le nom (ce qui est le plus souvent le cas), et permet également de faire référence à des champs sans nom, comme par exemple un champ *SUM(nomchamp)*. Notez toutefois qu'il est possible d'affecter un nom (ex. : *somme*) à ce genre de champ en indiquant `SUM(nomchamp) AS somme`.

L'exemple précédent gagnera ainsi en lisibilité et deviendra :

Listing 10.32 : select_eea_bd_inc.php

```
<?php

    /**
     * Fonction listant le contenu d'une table
     * contenant 2 champs (filmId et film)
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la table
     **/
    function EX_listeContenu($idConnexion, $table)
    {
```

```
        // Requete
        $requete = "SELECT * FROM $table";
        $idResultat = mysql_query($requete, $idConnexion);
        if (!$idResultat) return FALSE;

        // Boucle de lecture (et d'affichage) des enregistrements
        while ($enreg = mysql_fetch_array($idResultat)) {
            echo "FilmId=".$enreg["filmId"]." ".
                 "Film =" .$enreg["film"]."<br />";
        }
        return TRUE;
    }
?>
```



REMARQUE

mysql_fetch_assoc()

Utiliser la fonction `mysql_fetch_array()` *avec l'option MYSQL_ASSOC revient à appeler une fonction appelée* `mysql_fetch_assoc()`.

Pour votre culture personnelle, sachez qu'il existe également une fonction (d'un intérêt discutable) qui permet de retourner l'enregistrement sous forme d'un objet possédant des variables internes portant les mêmes noms (en minuscules) que les champs. Cette fonction s'appelle `mysql_fetch_object()` et possède la même syntaxe que `mysql_fetch_row()`, si ce n'est qu'elle retourne un objet et non un tableau.

La boucle de lecture se transforme alors en :

```
<?php
while ($enreg = mysql_fetch_object($idResultat)) {
    echo "FilmId = ".$enreg->filmid." Film = ".$enreg->film."<br />";
}
?>
```

Libération des ressources

Si, au cours d'un script, vous faites appel à de nombreuses requêtes retournant de nombreux enregistrements, alors, n'hésitez pas, une fois le résultat de la requête exploité, à libérer les ressources associées avec la fonction `mysql_free_result()`.

mysql_free_result()

Libère les ressources allouées pour un résultat de requête.

Syntaxe	`boolean mysql_free_result(resource $idResultat)`
`$idResultat`	Identifiant de requête tel que retourné par `mysql_query()`.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

Nombre d'enregistrements

Nombre d'enregistrements retournés

Il existe trois façons de déterminer le nombre d'enregistrements retournés par une requête.

Si vous souhaitez uniquement connaître le nombre d'enregistrements, mais ne souhaitez pas immédiatement lire ces enregistrements, alors il vous suffit de faire une requête `COUNT()` comme suit :

Listing 10.33 : count_bd_inc.php

```
<?php

    /**
     * Fonction affichant le nombre d'enregistrements
     * dans une table
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la tabme
     **/
    function EX_compte($idConnexion, $table)
    {
        // Requête
        $requete = "SELECT COUNT(*) FROM $table";
        $idResultat = mysql_query($requete, $idConnexion);

        // Lecture du résultat
        if ($enreg = mysql_fetch_row($idResultat)) {
            echo "Il y a ".$enreg[0]." enregistrements dans la table.<br />";
            return TRUE;
        } else {
            echo "Etes vous sûr que la table $table existe ?<br />";
            return FALSE;
        }
    }
?>
```

Si vous souhaitez connaître le nombre d'enregistrements, et que vous souhaitez lire les enregistrements sans pour autant avoir besoin au préalable d'en connaître le nombre, il suffit de compter combien d'enregistrements sont lus.

Listing 10.34 : count_select_db_inc.php

```
<?php

    /**
     * Fonction listant le contenu d'une table
     * contenant 2 champs (filmId et film)
     *
     * @param $idConnexion resource Identifiant de connexion BD
```

```
     * @param $table       string   Nom de la table
     **/
    function EX_listeContenu($idConnexion, $table)
    {
        // Requete
        $requete = "SELECT * FROM $table";
        $idResultat = mysql_query($requete, $idConnexion);
        if (!$idResultat) return FALSE;

        // Boucle de lecture (et de comptage) des enregistrements
        $nb = 0;
        while ($row = mysql_fetch_row($idResultat)) {
            // Vous pouvez manipuler $enreg à votre guise
            //echo "FilmId=".$enreg[0]." ".
            //     "Film =" .$enreg[1]."<br />";
            $nb++;
        }
        echo "Il y a $nb enregistrements dans la table.<br />";

        return TRUE;
     }
?>
```

Et, pour finir, ce que vous attendez tous : si vous souhaitez connaître le nombre d'enregistrements avant de les lire, vous pouvez faire appel à la fonction `mysql_num_rows()`.

mysql_num_rows()

Retourne le nombre d'enregistrements retournés par la requête SQL.

Syntaxe	`int mysql_num_rows(resource $idRequete)`
`$idRequete`	Identifiant de requête tel que retourné par `mysql_query()`.
`retour`	Nombre d'enregistrements, ou rien en cas d'identifiant de requête non valide.

Listing 10.35 : count_num_rows_bd_inc.php

```
<?php

    /**
     * Fonction listant le contenu d'une table
     * contenant 2 champs (filmId et film)
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la table
     **/
    function EX_listeContenu($idConnexion, $table)
    {
```

```
        // Requete
        $requete = "SELECT * FROM $table";
        $idResultat = mysql_query($requete, $idConnexion);
        if (!$idResultat) return FALSE;

        echo "Il y a ".mysql_num_rows($idResultat)." enregistrements ".
             "dans la table<br />";

        return TRUE;
    }
?>
```

Sachant qu'une requête `SELECT *` est plus longue qu'une requête `SELECT COUNT(*)`, privilégiez la première méthode si vous n'avez que faire du contenu des enregistrements.

Nombre d'enregistrements modifiés

Pour connaître le nombre de lignes affectées par la requête précédente (comme le nombre de résultats après un `SELECT`, le nombre de lignes effacées après un `DELETE` ou le nombre de lignes modifiées après un `UPDATE`), nous pouvons utiliser la fonction `mysql_affected_rows()`.

mysql_affected_rows()

Permet de retrouver le nombre de lignes affectées par la requête immédiatement précédente.

Syntaxe	`int mysql_affected_rows([resource $idConnexion])`
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `mysql_connect()`. Par défaut, c'est l'identifiant de la dernière connexion qui est utilisé.

Bien que plus difficilement exploitable, sachez que depuis PHP 4.3, vous diposez également de la fonction `mysql_info()`.

mysql_info()

Retourne une chaîne de caractères précisant le nombre d'enregistrements ajoutés, modifiés, etc. durant la dernière requête. Toutefois, ceci ne concerne qu'un nombre réduit de type de requêtes (INSERT avec des valeurs issues d'un SELECT, INSERT de plusieurs lignes, LOAD DATA INFILE, ALTER TABLE et UPDATE) en fait il s'agit de celles susceptible de modifier plusieurs enregistrements à la fois.

Syntaxe :	`string mysql_info([ressource $idConnexion])`
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `mysql_connect()`. Par défaut, c'est l'identifiant de la dernière connexion qui est utilisé.

`retour`	Chaîne de caractères semblable à "Records: 2 Duplicates: 0 Warnings: 0" ou `FALSE` si la requête ne retourne pas d'info.

Gestion des erreurs

Jusque-là, en cas d'erreur, nous nous sommes contentés d'afficher un message générique. Il est cependant possible de déterminer plus précisément l'origine de l'erreur.

Cela est notamment possible en récupérant un code d'erreur via la fonction `mysql_errno()`.

mysql_errno()

Retourne le code d'erreur du dernier appel MySQL.

Syntaxe	`int mysql_errno([resource $idConnexion])`
`$idConnexion`	Identifiant de connexion au serveur de bases de données tel que retourné par `mysql_connect()` ou `mysql_pconnect()`. Par défaut, c'est l'identifiant de la dernière connexion qui est utilisé.
`retour`	Code d'erreur, ou tout simplement 0 si la dernière opération s'est passée sans erreur.

mysql_error()

Retourne le message d'erreur associé au dernier appel MySQL. Ce message dépend du serveur, mais aura toutes les chances d'être en anglais.

Syntaxe	`string mysql_error([resource $idConnexion])`
`$idConnexion`	Identifiant de connexion au serveur de bases de données tel que retourné par `mysql_connect()` ou `mysql_pconnect()`. Par défaut, c'est l'identifiant de la dernière connexion qui est utilisé.
`retour`	Message d'erreur, ou tout simplement une chaîne vide "" si la dernière opération s'est passée sans erreur.

Tableau 10.14 : Exemples de codes et messages d'erreur

Code	Message	Description
0	''	Pas d'erreur
1049	Unknown database 'nombase'	La base `'nombase'` n'existe pas
1051	Unknown table 'nomtable'	La table `'nomtable'` n'existe pas
1054	Unknown column 'nomchamp' in 'field list'	Le champ `'nomchamp'` précisé dans la liste de champs n'existe pas

Code	Message	Description
1054 (et oui ! le même code que précédemment)	Unknown column 'nomchamp' in 'where clause'	Le champ `'nomchamp'` précisé dans la clause WHERE n'existe pas
2005	Unknown MYSQL Server Host 'nomserveur' (2)	Le serveur MySQL `'nomserveur'` n'existe pas

Vous pourrez, par exemple, vous servir des codes d'erreur pour afficher vos propres messages d'erreur. Dans l'exemple suivant, nous montrons comment traduire les messages d'erreur (cela ne s'appliquera qu'à l'échantillon de codes qui vient d'être présenté).

Listing 10.36 : mysql_err_inc.php

```
<?php

    // Retourne une version Française du message d'erreur
    // en se basant sur le code d'erreur et sur les informations
    // complémentaires que l'on peut trouver dans le message
    // Anglais
    function mysql_error_fr()
    {
      // Les noms de tables et autres informations
      // sont entre apostrophes dans le message d'erreur
      // il est donc utile de le décomposer
      $morceaux = explode("'", mysql_error());

      switch (mysql_errno()) {
         case 0 : // message vide
             return "";

         case 1049 : // Unknown database 'nombase'
             return "La base '".$morceaux[1]."' n'existe pas";

         case 1051 : // Unknown table 'nomtable'
             return "La table '".$morceaux[1]."' n'existe pas";

         case 1054 : // Unknown column 'nomchamp' in 'field list'
                     // Unknown column 'nomchamp' in 'where clause'
             switch ($morceaux[3]) {
                 case 'field list' :
                     return "Le champ '".$morceaux[1]."' précisé".
                            " dans la liste des champs n'existe pas";
                 case 'where clause' :
                     return "Le champ '".$morceaux[1]."' précisé".
                            " dans la clause WHERE n'existe pas";
                 default :
                    return mysql_error();
             }
         case 2005 : // Unknown MYSQL Server Host 'nomserveur' (2)
                 return "Le serveur MySQL '".$morceaux[1]."' n'existe pas";
```

```
            default :
                return mysql_error();
        }
    }
?>
```

Et voici un script générant l'ensemble des erreurs ci-dessus, afin de bien mettre en évidence le bon fonctionnement de cette fonction personnalisée :

Listing 10.37 : mysql_err01.php

```
<?php

    include_once("parametres_bd_inc.php");
    include_once("mysql_err_inc.php");

    $table = "tableerr";

    @mysql_pconnect("erreur");
    echo mysql_error_fr()."<br />";

    mysql_pconnect($serveur.$port, $utilisateur, $motDePasse) or
        die("<b>Impossible de se connecter au serveur base de données.</b>");

    @mysql_select_db("erreur");
    echo mysql_error_fr()."<br />";

    mysql_select_db($base) or
        die("<b>Grrr... J'aurais aimé qu'elle existe cette base.</b>");

    @mysql_query("DROP TABLE erreur");
    echo mysql_error_fr()."<br />";

    mysql_query("CREATE TABLE IF NOT EXISTS $table (id INT2)") or
        die("<b>Grrr... J'aurais aimé que cette requête ne tombe".
            " pas en erreur.</b>");

    @mysql_query("SELECT erreur FROM $table");
    echo mysql_error_fr()."<br />";

    @mysql_query("SELECT * FROM $table WHERE erreur=1");
    echo mysql_error_fr()."<br />";

?>
```

Ce script génère alors une page contenant les messages :

```
Le serveur MySQL 'erreur' n'existe pas
La base 'erreur' n'existe pas
La table 'erreur' n'existe pas
Le champ 'erreur' précisé dans la liste des champs n'existe pas
Le champ 'erreur' précisé dans la clause WHERE n'existe pas
```

Exemples d'applications

Compteur de clics

Une application courante d'utilisation des bases de données, et simple à mettre en œuvre, consiste à créer un compteur de clics.

En effet, peut-être souhaitez-vous proposer, sur votre site, un certain nombre de liens vers des sites Internet (annuaire web) ou vers des fichiers (bibliothèques de scripts). Peut-être que certains (ou la totalité) de ces liens sont proposés par un partenaire. Bref, tout cela vous incite à savoir quelles sont les pages les plus fréquemment consultées (pour connaître les centres d'intérêt des visiteurs) et combien de visiteurs vous avez redirigés vers votre sponsor.

La méthode la plus classique consiste à stocker en base de données l'ensemble des liens ainsi proposés, et à remplacer les traditionnels liens de la forme **http://www.domaine.com** par un lien vers un script chargé d'incrémenter le compteur correspondant au site indiqué, et de rediriger le client vers le site grâce à la fonction `header()`.


RENVOI

Vous pouvez vous reporter à l'annexe "Les en-têtes" pour plus de renseignements sur la redirection.

Pour notre exemple, nous utiliserons une table appelée *url* contenant trois champs :

- *urlid*, identifiant de l'URL (champ auto-incrémenté) ;
- *url*, l'URL proprement dite ;
- *nbclic*, le nombre de clics.

Le script de redirection (appelé ici *compteurclic_redirection.php*) devra alors accepter un paramètre (`urlid`) et devra, à partir de ce paramètre, déterminer quelle est la valeur du champ *url*, incrémenter *nbclic* pour cette URL, et rediriger vers *url*.

Les appels à ce script auront alors la forme **http://localhost/compteurclic_redirection.php?urlid=3** (pour accéder à l'URL ayant l'identifiant 3). À supposer que l'URL 3 corresponde au site **http://www.sqlfacile.com** cela signifie que le lien `<a href="http://www.sqlfacile.com ">` devra être remplacé par `<a href="compteurclic_redirection.php?urlid=3>` si l'on souhaite en compter le nombre de clics.

Concrètement, cette table pourra être créée et alimentée avec la fonction `CC_initialiseBD()` du script suivant :

Listing 10.38 : compteurclic_bd_inc.php (début)

```
<?php
    //-----------------------------------
    // Charge les paramètres de connexion
    // à la base de données.
    //-----------------------------------
    require_once("parametres_bd_inc.php");
```

```
/**
 * Fonction de connexion à une base de données
 * s'appuie sur les paramètres fournis
 * dans parametres_bd_inc.php.
 *
 * @return resource Identifiant de connexion
 */
function CC_connexion()
{
    global $serveur, $base, $utilisateur, $motDePasse;

    $idConnexion = mysql_pconnect($serveur, $utilisateur, $motDePasse);
    if (!$idConnexion) return FALSE;
    mysql_select_db($base);

    return $idConnexion;
}

/**
 * Fonction de deconnexion.
 * (Ne fait rien sachant que la fonction
 * de connexion établit une connexion persistante)
 */
function CC_deconnexion()
{
    return TRUE;
}

/**
 * Fonction chargé de créer et d'alimenter
 * la table "compteur de clics"
 **/
function CC_initialiseBD($idConnexion, $table)
{
    // Création de la table
    $requete = "DROP TABLE $table";
    @mysql_query($requete, $idConnexion);

    $requete = "CREATE TABLE $table (".
                                "urlId INTEGER ".
                                "AUTO_INCREMENT PRIMARY KEY,".
                                "url VARCHAR(128) NOT NULL,".
                                "nbclic INTEGER DEFAULT 0,".
                                "UNIQUE(url))";
    if (!mysql_query($requete, $idConnexion)) return FALSE;

    // Alimentation de la table
    $requete = "INSERT INTO $table (url)".
                    " VALUES ('http://www.php.net')";
    if (!mysql_query($requete, $idConnexion)) return FALSE;

    $requete = "INSERT INTO $table (url)".
```

```
                    " VALUES ('http://www.phpfacile.com')";
        if (!mysql_query($requete, $idConnexion)) return FALSE;

        $requete = "INSERT INTO $table (url)".
                    " VALUES ('http://www.sqlfacile.com')";
        if (!mysql_query($requete, $idConnexion)) return FALSE;

        $requete = "INSERT INTO $table (url)".
                    " VALUES ('http://www.xmlfacile.com')";
        if (!mysql_query($requete, $idConnexion)) return FALSE;

        $requete = "INSERT INTO $table (url)".
                    " VALUES ('http://www.ootoogo.com')";
        if (!mysql_query($requete, $idConnexion)) return FALSE;

        return TRUE;
    }
?>
```

La lecture de la liste des liens disponibles en base de données pourra se faire via la fonction `CC_recupereLiens()` du script suivant :

Listing 10.39 : compteurclic_bd_inc.php (milieu)

```
<?php
    /**
     * Fonction retournant les informations de liens
     * sous forme d'un tableau associatif possedants
     * les clés
     * - "lien" associé au tableau des liens hypertextes
     * - "nbclic" associé au tableau des nombres de clics
     **/
    function CC_recupereLiens($idConnexion, $table)
    {
        // Nom du script chargé du comptage et de la redirection
        $script = "compteurclic_redirection.php";

        // Requête SELECT
        $requete = "SELECT * FROM $table";
        $idResultat = mysql_query($requete, $idConnexion);

        // Récupération des enregistrements les uns après les autres
        while ($enreg = mysql_fetch_array($idResultat)) {
            $liens["lien"][] = "<a href=\"$script?urlid=".
                               $enreg["urlId"]."\">".
                               $enreg["url"]."</a>";
            $liens["nbclic"][] = $enreg["nbclic"];
        }

        return $liens;
    }
```

L'affichage des liens consiste uniquement à mettre en page le contenu du tableau ainsi récupéré (voir la fonction du script *compteurclic_html_inc.php*).

Comme cela a été précisé, le lien `<a href="http://www.sqlfacile.com">http://www.sqlfacile.com</a>` a été remplacé par `<a href="compteurclic_redirection.php?urlid=3">http://www.sqlfacile.com </a>`. La redirection vers le site **www.sqlfacile.com** est donc à la charge du script *compteurclic_redirection.php*.

Listing 10.40 : compteurclic_redirection.php

```
<?php

    // Paramètres du script
    require_once("compteurclic_bd_inc.php");
    $table = "compteurclic";

    // Connexion à la base de données
    $idConnexion = @CC_connexion();
    if (!$idConnexion) {
        die("<b>Impossible de se connecter à la base de données</b>");
    }

    // Récupération de l'url et incrémentation du compteur
    // pour l'url d'indentifiant passé en paramètre de ce script
    // par la méthode GET
    $url = @CC_recupereUrl($idConnexion, $_GET["urlid"]);

    // Deconnexion
    @CC_deconnexion();

    // C'est l'heure de la redirection
    if ($url) {
        header("Location: $url");
    } else {
        echo "Désolé, nous ne pouvons vous proposer ce lien";
    }
?>
```

Ce script appelle principalement la fonction `CC_recupereUrl()` suivante :

Listing 10.41 : compteurclic_bd_inc.php (fin)

```
<?php
/**
     * Fonction récupérant une url à partir de son identifiant
     * et incrémentant le compteur de clics
     **/
    function CC_recupereUrl($idConnexion, $urlid)
    {
        global $table;

        // Récupère l'url
```

```
        $requete = "SELECT * FROM $table WHERE urlid=$urlid";
        $idResultat = mysql_query($requete, $idConnexion);

        if ($enreg = mysql_fetch_array($idResultat)) {
            $url = $enreg["url"];

            // Incrémente le compteur
            $requete = "UPDATE $table SET nbclic=nbclic+1 WHERE urlid=$urlid";
            mysql_query($requete, $idConnexion);
        } else {
            $url = FALSE;
        }
        return $url;
    }
?>
```


REMARQUE

Utilisation du bouton retour

Si, après avoir suivi un lien, vous revenez sur la page compteurclic.php en utilisant le bouton retour (back), le contenu de la page ne sera pas réactualisé. Par conséquent, le nombre de visites affiché ne sera pas correct. N'oubliez donc pas, dans ce cas, de rafraîchir (bouton actualiser) la page.

SuperTheque

L'essentiel du code de l'application a été présenté en introduction de ce chapitre. Il ne restait plus qu'à connaître les fonctions MySQL pour implémenter l'interface `RessourceInterface` c'est désormais chose faite:

Listing 10.42 : RessourceMySQL_class.php

```
<?php
include_once("RessourceInterface_class.php");
include_once(dirname(__FILE__)."/../config/mysql_cfg.php");
/**
 * RessourceMySQL_class.php
 * Classe d'acces a une base de donnees MySQL
 * Cette classe doit implementer toutes les methodes de l'interface
 *   RessourceInterface.
 * Compatibilite: PHP 5
 */
class Ressource implements RessourceInterface
{
    var $idConnexion;

    public function connexion()
    {
        global $mysql_serveur, $mysql_utilisateur, $mysql_motDePasse;
```

```
    global $mysql_base;

    if (!isset($this->idConnexion)) {
        $idConnexion = mysql_pconnect($mysql_serveur,
                                      $mysql_utilisateur,
                                      $mysql_motDePasse);
        if (!$idConnexion) return FALSE;
        mysql_select_db($mysql_base);
        $this->idConnexion = $idConnexion;
    }
    return TRUE;
}

public function deconnexion()
{
    // Rien a faire, il s'agit d'une connexion persistante
}

public function reset()
{
    $requete = "DROP TABLE types";
    $idResultat = mysql_query($requete, $this->idConnexion);
    // Pas de raison de retourner un erreur
    // si la suppression echoue (si la table n'existe pas)

    $requete = "CREATE TABLE types (".
                    "id INTEGER AUTO_INCREMENT PRIMARY KEY,".
                    "type VARCHAR(255))";
    $idResultat = mysql_query($requete, $this->idConnexion);
    if (!$idResultat) return FALSE;

    $types = array("Album", "Film", "Livre", "Musique");
    foreach ($types as $type) {
        $requete = "INSERT INTO types (type) VALUES ('$type')";
        $idResultat = mysql_query($requete, $this->idConnexion);
        if (!$idResultat) return FALSE;
    }

    $requete = "DROP TABLE articles";
    $idResultat = @mysql_query($requete, $this->idConnexion);
    // Pas de raison de retourner un erreur
    // si la suppression echoue (si la table n'existe pas)

    $requete = "CREATE TABLE articles (".
                "id INTEGER AUTO_INCREMENT PRIMARY KEY,".
                "albumId INTEGER,".
                "titre VARCHAR(255),".
                "typeId INTEGER,".
                "commentaire VARCHAR(255))";
    $idResultat = mysql_query($requete, $this->idConnexion);
    if (!$idResultat) return FALSE;
}
```

```
public function addArticle($albumId,
                           $titre,
                           $typeId,
                           $commentaire)
{
    $requeteDebut = "INSERT INTO articles (albumId, titre, typeId";
    $requeteFin = ") VALUES ($albumId,".
                            "'".mysql_escape_string($titre)."',".
                            "$typeId";
    if ($commentaire != "") {
         $requeteDebut .= ",commentaire";
         $requeteFin .= ",'".mysql_escape_string($commentaire)."'";
    }
    $requete = $requeteDebut . $requeteFin . ")";
    $idResultat = mysql_query($requete, $this->idConnexion);
    if (!$idResultat) return FALSE;
}

public function getArticle($articleId)
{
    $requete = "SELECT * FROM articles WHERE id=$articleId";
    $idResultat = mysql_query($requete, $this->idConnexion);
    if (!$idResultat) return FALSE;

    $article = NULL;
    if ($enreg = mysql_fetch_array($idResultat)) {
        $article = $this->enreg2Article($enreg);
    }
    return $article;
}

public function getArticles(Filtre $filtre, Plage $plage, Tri $tri)
{
    $requete = "SELECT * FROM articles";
    if ($filtre->getAlbumId() != -1) {
        $conditionSQL = "albumId=".$filtre->getAlbumId();
    }
    if ($filtre->getTitre() != "") {
        $conditionSQL = $conditionSQL.
                        " AND titre LIKE '".
                        mysql_escape_string($filtre->getTitre())."'";
    }
    if ($filtre->getTypeId() != -1) {
        $conditionSQL = $conditionSQL.
                        " AND typeId=".$filtre->getTypeId();
    }

    if ($tri->getSens() == -1) {
        $triSQL = "ORDER BY ".$tri->getChamp()." DESC";
    } else if ($tri->getSens() == 1) {
        $triSQL = "ORDER BY ".$tri->getChamp()." ASC";
```

```
    }

    $limiteSQL = "LIMIT ".$plage->getNbArticleMax().
                 " OFFSET ".$plage->getPremierArticle();

    $requete = $requete." WHERE ".$conditionSQL." ".
                                  $triSQL." ".$limiteSQL;
    $idResultat = mysql_query($requete, $this->idConnexion);
    if (!$idResultat) return FALSE;

    $articles = NULL;
    while ($enreg = mysql_fetch_array($idResultat)) {
        $articles[] = $this->enreg2Article($enreg);
    }
    return $articles;
}

public function getNbTotalArticles(Filtre $filtre)
{
    $requete = "SELECT COUNT(*) FROM articles";
    if ($filtre->getAlbumId() != -1) {
        $conditionSQL = "albumId=".$filtre->getAlbumId();
    }
    if ($filtre->getTitre() != "") {
        $conditionSQL = $conditionSQL.
                        " AND titre LIKE '".
                        mysql_escape_string($filtre->getTitre())."'";
    }
    if ($filtre->getTypeId() != -1) {
        $conditionSQL = $conditionSQL.
                        " AND typeId=".$filtre->getTypeId();
    }

    $requete = $requete." WHERE ".$conditionSQL;
    $idResultat = mysql_query($requete, $this->idConnexion);
    if (!$idResultat) return FALSE;

    if ($enreg = mysql_fetch_array($idResultat)) {
        return $enreg[0];
    } else return FALSE;
}

public function getTypes()
{
    $requete = "SELECT * FROM types";
    $idResultat = mysql_query($requete, $this->idConnexion);
    if (!$idResultat) return FALSE;

    $types = NULL;
    while ($enreg = mysql_fetch_array($idResultat)) {
        $types[$enreg["id"]] = $enreg["type"];
    }
```

```
        return $types;
    }

    public function getAlbumTypeId()
    {
         return 1;
    }


    private function enreg2Article($enreg)
    {
        $article = new Article();
        $article->setId($enreg["id"]);
        $article->setAlbumId($enreg["albumId"]);
        $article->setTitre($enreg["titre"]);
        $article->setTypeId($enreg["typeId"]);
        $article->setCommentaire($enreg["commentaire"]);
        return $article;
    }

}
?>
```

Comme vous le constatez, la principale difficulté n'est pas tant au niveau des fonctions disponibles qu'au niveau de la construction des requêtes SQL. La requête sera simple lorsqu'il ne s'agit que de récupérer un enregistrement identifié par la clé `Id` (comme c'est le cas avec `getArticle()`) elle sera bien plus complexe s'il s'agit des récupérer un ensemble d'enregistrements répondant à des critères précis et triés (comme c'est le cas avec `getArticles()`).

Requêtes avec des apostrophes

Lors de la construction de la requête, il faut bien garder à l'esprit que l'un des éléments de la requête (typiquement un champ de type texte saisi par l'utilisateur, comme ici le titre ou le commentaire) peut contenir une apostrophe qui pourrait être confondu avec le délimiteur de chaîne. Pensez donc bien à faire un appel à `addSlashes()` lorsque cela peut s'avérer nécessaire.

Affichage page par page

La meilleure façon avec MySQL pour n'extraire qu'un sous-ensemble des enregistrements retournés par une requête SQL consiste à utiliser l'instruction `LIMIT`. `LIMIT` permet de préciser combien d'enregistrements doivent être retournés et à partir duquel commencer (sachant que le premier enregistrement porte l'index 0).

Ainsi, `SELECT * FROM nomtable LIMIT 0, 10;` retourne les dix premiers enregistrements (de l'index 0 à l'index 9) et `SELECT * FROM nomtable LIMIT 10, 20;` retourne les vingt suivants (de l'index 10 à l'index 29).

Tri

Modifier l'ordre d'affichage des annonces n'est pas non plus bien sorcier. Le tri peut s'effectuer directement au niveau de la requête SQL, grâce à l'instruction `ORDER BY`.

Moteur de recherche (filtre)

Pour filtrer, il suffit d'utiliser l'instruction `WHERE`. Dans notre cas, nous souhaitons autoriser l'utilisation de jokers (comme %) afin, par exemple, de rechercher les titres commençant par une chaîne donnée. C'est pourquoi, nous ne ferons pas un test de type `titre='motcle'` mais `titre LIKE 'motcle'` (où `motcle` pourrait avoir la valeur "La 7eme compagnie%").

En savoir plus...

… sur une requête

Si vous souhaitez connaître le nombre de champs retournés par une requête, faites simplement appel à `mysql_num_fields()`.

mysql_num_fields()

Retourne le nombre de champs d'un enregistrement.

Syntaxe	`int mysql_num_fields(resource $idRequete)`
`$idRequete`	Identifiant de requête tel que retourné par `mysql_list_tables()`.
`retour`	Nombre de champs retournés par une requête.

… sur les champs d'une table

Vous pouvez récupérer la liste des champs contenus dans une base en lançant une requête avec `mysql_list_fields()`, et en lisant le résultat avec `mysql_field_name()`.

mysql_list_fields()

Retourne un identifiant sur une requête listant les champs disponibles dans la table.

Syntaxe	`resource mysql_list_fields(string $nomBaseDonnees, string $nomTable, [resource $idConnexion])`
`$nomBaseDonnees`	Nom de la base de données contenant la table dont on souhaite connaître les champs.
`$nomTable`	Nom de la table dont on souhaite connaître les champs.

`$idConnexion`	Identifiant de connexion à un serveur de bases de données tel que retourné par `mysql_connect()` ou `mysql_pconnect()`. Par défaut, c'est l'identifiant de la dernière connexion qui est utilisé.
`retour`	Identifiant de la requête, ou `FALSE` en cas d'échec (nom de base ou de table non valide).

mysql_field_name()

Retourne le nom des champs dans la table.

Syntaxe	`string mysql_field_name(resource $idRequete, int $index)`
`$idRequete`	Identifiant de requête tel que retourné par `mysql_list_tables()`.
`$index`	Index du champ dont on souhaite connaître le nom (le premier champ porte l'index 0). Le nombre total de champs peut être obtenu en faisant appel à `mysql_num_fields()`.
`retour`	Nom du champ, ou `FALSE` si l'index n'est pas valide, ou rien si l'identifiant de résultat n'est pas valide.

Cependant, les champs ont d'autres informations à nous livrer que leur nom. Pour cela, vous diposez des fonctions :

mysql_field_type()

Retourne le type des champs de la table.

Syntaxe	`string mysql_field_type(resource $idRequete, int $index)`
`$idRequete`	Identifiant de requête tel que retourné par `mysql_list_tables()`.
`$index`	Index du champ dont on souhaite connaître le type. Le nombre total de champs peut être obtenu en faisant appel à `mysql_num_fields()`.
`retour`	Type du champ, ou `FALSE` en cas d'erreur.

mysql_field_len()

Retourne la taille (en octets) des champs de la table.

Syntaxe	`string mysql_field_len(resource $idRequete, int $index)`
`$idRequete`	Identifiant de requête tel que retourné par `mysql_list_tables()`.
`$index`	Index du champ dont on souhaite connaître le type. Le nombre total de champs peut être obtenu en faisant appel à `mysql_num_fields()`.

retour	Taille en octets occupée par le champ, ou FALSE en cas d'erreur d'index ou rien en cas d'identifiant de requête non valide.

mysql_field_flags()

Retourne les attributs (NOT NULL, etc.) des champs de la table.

Syntaxe	string mysql_field_flags(resource $idRequete, int $index)
$idRequete	Identifiant de requête tel que retourné par mysql_list_tables().
$index	Index du champ dont on souhaite connaître le type. Le nombre total de champs peut être obtenu en faisant appel à mysql_num_fields().
retour	Chaîne de caractères contenant les attributs séparés par des espaces (les blancs contenus dans les attributs sont remplacés par des underscores (ex : not_null) ; FALSE en cas d'erreur d'index ou rien en cas d'identifiant de requête non valide.

… sur les tables d'une base

Le principe est exactement le même pour récupérer la liste des tables contenues dans une base.

mysql_list_tables()

Retourne un identifiant sur une requête listant les tables disponibles dans la base de données.

Syntaxe	resource mysql_list_tables(string $nomBaseDonnees, [resource $idConnexion])
$nomBaseDonnees	Nom de la base de données dont on souhaite connaître les tables.
$idConnexion	Identifiant de connexion à un serveur de bases de données tel que retourné par mysql_connect() ou mysql_pconnect(). Par défaut, c'est l'identifiant de la dernière connexion qui est utilisé.
retour	Identifiant de la requête, ou FALSE en cas d'échec (nom de base non valide).

mysql_tablename()

Retourne le nom des tables dans la base.

Syntaxe	string mysql_tablename(resource $idRequete, int $index)
$idRequete	Identifiant de requête tel que retourné par mysql_list_tables().

`$index`	Index de la table dont on souhaite connaître le nom. Le nombre total de tables peut être obtenu en faisant appel à `mysql_num_rows()`.
`retour`	Nom de la table, ou `FALSE` en cas d'erreur d'index, ou rien en cas d'identifiant de requête non valide.

… sur les bases d'un serveur

mysql_list_dbs()

Retourne un identifiant sur une requête listant les bases de données disponibles sur le serveur.

Syntaxe	`resource mysql_list_dbs([resource $idConnexion])`
`$idConnexion`	Identifiant de connexion à un serveur de bases de données tel que retourné par `mysql_connect()` ou `mysql_pconnect()`. Par défaut, c'est l'identifiant de la dernière connexion qui est utilisé.
`retour`	Identifiant de la requête.

mysql_db_name()

Retourne le nom des bases de données sur le serveur.

Syntaxe	`string mysql_db_name(resource $idRequete, int $index)`
`$idRequete`	Identifiant de requête tel que retourné par `mysql_list_dbs()`.
`$index`	Index de la base de données dont on souhaite connaître le nom. Le nombre total de bases peut être obtenu en faisant appel à `mysql_num_rows()`.
`retour`	Nom de la base de données, ou `FALSE` en cas d'erreur (ex. : index non valide), voire rien pour d'autres cas d'erreur (ex. : identifiant de requête non valide).

Vous pourrez donc lister les champs de la première table de la première base de votre serveur avec le script suivant :

Listing 10.43 : mysql_13.php

```
<?php

    include_once("parametres_bd_inc.php");

    mysql_pconnect($serveur.$port, $utilisateur, $motDePasse) or
        die("<b>Impossible de se connecter au serveur base de données.</b>");

    $idListeBD = mysql_list_dbs();
```

```
    if ($nomBD = @mysql_db_name($idListeBD, 0)) {
       //$nomBD = "basemysql";  // Pour forcer un nom de base
       echo "Votre serveur contient au moins la base <b>$nomBD</b><br />";

       $idListeTable = mysql_list_tables($nomBD);
       if ($nomTable = @mysql_tablename($idListeTable, 0)) {
           echo "Et elle continent au moins la table <b>$nomTable</b><br />";
           echo "Dont les champs sont:<br />";
           $idListeChamp = mysql_list_fields($nomBD, $nomTable);

           echo "Rows=".mysql_num_rows($idListeChamp)."<br />";
           echo "Col=".mysql_num_fields($idListeChamp)."<br />";

           $nb = 0;
           while ($nomChamp = @mysql_field_name($idListeChamp, $nb)) {
               $type     = mysql_field_type($idListeChamp, $nb);
               $taille   = mysql_field_len($idListeChamp, $nb);
               $attribut = mysql_field_flags($idListeChamp, $nb);
               echo "- <i>$nomChamp</i> de type <i>$type</i>";
               echo " occupant <i>$taille</i> octets";
               echo " avec les attributs <i>$attribut</i><br />";
               $nb++;
           }
       }
    } else {
       die("Il semblerait que votre serveur ne contienne aucune base");
    }
?>
```

… sur la connexion courante

PHP 4.3 a apporté son lot de nouvelles fonctions permettant notamment d'en savoir un peu plus sur la connexion courante: Est-elle encore active? Quel identifiant lui est associé?

mysql_ping()

Teste la connexion avec le serveur de base de données et tente de la rétablir si elle a été perdue (apr exemple sur délai expiré).

Syntaxe :	`boolean mysql_ping([resource $idConnexion])`
`$idConnexion`	Identifiant de connexion à un serveur de bases de données tel que retourné par `mysql_connect()` ou `mysql_pconnect()`. Par défaut, c'est l'identifiant de la dernière connexion qui est utilisé. S'il n'y a pas eu de précédente tentative de connexion, c'est la connexion locale qui est testée.
`retour`	`TRUE` si la connexion est toujours établie, `FALSE` sinon.

mysql_thread_id()

Retourne l'identifiant de la connexion (thread) sur le serveur de bases de données.

Syntaxe :	`int mysql_thread_id([resource $idConnexion])`
`$idConnexion`	Identifiant de connexion à un serveur de bases de données tel que retourné par `mysql_connect()` ou `mysql_pconnect()`. Par défaut, c'est l'identifiant de la dernière connexion qui est utilisé.
`retour`	Identifiant de connexion sur le serveur de base de données.

… sur l'ensemble des connexions

Egalement apparues depuis la version 4.3 de PHP, les fonctions suivantes:

mysql_stat()

Retourne diverses informations (nombre de connexions ouvertes, nombre de requêtes traitées, nombre de tables ouvertes, etc.) sur le serveur de bases de données.

Syntaxe :	`string mysql_stat([resource $idConnexion])`
`$idConnexion`	Identifiant de connexion à un serveur de bases de données tel que retourné par `mysql_connect()` ou `mysql_pconnect()`. Par défaut, c'est l'identifiant de la dernière connexion qui est utilisé.
`retour`	Chaîne de caractères contenant l'ensemble des informations disponibles, sous une forme semblable à "Uptime: 22 Threads: 1 Questions: 9 Slow queries: 0 Opens: 7 Flush tables: 1 Open tables: 1 Queries per second avg: 0.409"

mysql_list_processes()

Retourne un identifiant sur une requête listant les connexions sur le serveur de bases de données.

Syntaxe :	`resource mysql_list_processes([resource $idConnexion]]`
`$idConnexion`	Identifiant de connexion à un serveur de bases de données tel que retourné par `mysql_connect()` ou `mysql_pconnect()`. Par défaut, c'est l'identifiant de la dernière connexion qui est utilisé.
`retour`	Identifiant de la requête.

… sur les numéros de version

Les fonctions évoquées ici ne sont pas d'une utilisation très courante, mais elles pourront éventuellement vous servir si vous êtes amené à utiliser des instructions avancées de la base de données MySQL (instructions implémentées dans des versions récentes, par exemple).

mysql_get_client_info()

Retourne le numéro de version de client MySQL intégré à PHP.

Syntaxe	`string mysql_get_client_info(void)`
`retour`	Numéro de version du client MySQL.

mysql_get_server_info()

Retourne le numéro de version du serveur MySQL sur lequel nous sommes connectés.

Syntaxe	`string mysql_get_server_info([resource $idConnexion])`
`$idConnexion`	Identifiant de connexion à un serveur de bases de données tel que retourné par `mysql_connect()` ou `mysql_pconnect()`. Par défaut, c'est l'identifiant de la dernière connexion qui est utilisé.
`retour`	Numéro de version du serveur.

mysql_get_host_info()

Retourne le nom du serveur MySQL sur lequel nous sommes connectés et le mode de connexion.

Syntaxe	`string mysql_get_host_info([resource $idConnexion])`
`$idConnexion`	Identifiant de connexion à un serveur de bases de données tel que retourné par `mysql_connect()` ou `mysql_pconnect()`. Par défaut, c'est l'identifiant de la dernière connexion qui est utilisé.
`retour`	Nom et mode de connexion au serveur.

mysql_get_proto_info()

Retourne le numéro de version du protocole de connexion utilisé.

Syntaxe	`string mysql_get_proto_info([resource $idConnexion])`
`$idConnexion`	Identifiant de connexion à un serveur de bases de données tel que retourné par `mysql_connect()` ou `mysql_pconnect()`. Par défaut, c'est l'identifiant de la dernière connexion qui est utilisé.
`retour`	Numéro de version du protocole

Ainsi, le script suivant :

Listing 10.44 : mysql_14.php

```
<?php

    include_once("parametres_bd_inc.php");

    mysql_pconnect($serveur.$port, $utilisateur, $motDePasse) or
        die("<b>Impossible de se connecter au serveur base de données.</b>");

    echo "Cette version de PHP intègre la version <b>".
         mysql_get_client_info();
    echo "</b> du client MySQL<br /><br />";

    echo "Et nous sommes connectés au serveur <b>".mysql_get_host_info();
    echo "</b><br />";
    echo "sur lequel tourne la version <b>".mysql_get_server_info()."</b>";
    echo "<br /><br />";

    echo "Nous utilisons (sans le savoir) la version ";
    echo "<b>".mysql_get_proto_info()."</b> du protocole de connexion.";
?>
```

retournera une page du genre (selon votre configuration) :

```
Cette version de PHP intègre la version 3.23.40 du client MySQL
Et nous sommes connectés au serveur Localhost via UNIX socket
sur lequel tourne la version 3.2.3.40
Nous utilisons (sans le savoir) la version 10 du protocole de connexion.
```

10.10. ODBC

Que vous utilisiez véritablement une liaison ODBC, ou que vous utilisiez une base de données comme DB2 (d'IBM), ce sont ces fonctions que vous utiliserez.

Installation

Si vous devez utiliser une base de données via une véritable liaison ODBC, vous devrez au préalable configurer un pont ODBC (si le cas de votre base de données n'est pas traité dans ce manuel, vous pouvez, à titre d'exemple, consulter le cas de la base de données MS Access).

Configuration php.ini

Le fichier *php.ini* permet de définir les paramètres suivants

`odbc.allow_persistent = On`

Autorise ou interdit les connexions persistantes.

`odbc.check_persistent = On`

Vérifie si la connexion est encore valide avant chaque réutilisation.

`odbc.max_persistent = -1`

Nombre maximum de connexions persistantes. -1 autorise un nombre illimité.

`odbc.max_links = -1`

Nombre maximum de connexions (persistantes ou non). -1 autorise un nombre illimité.

`odbc.defaultlrl = 4096`

Indique le nombre d'octets qui doivent être retournés dans le cas des champs longs. La valeur 0 permet de passer outre.

`odbc.defaultbinmode = 1`

Précise comment passer les données binaires : 1 retourne les données telles quelles, 2 convertit les données en caractères.

Utilisation

L'utilisation basique de la base de données s'opère selon le schéma suivant.

- Connexion grâce à la fonction `odbc_connect()` ou `odbc_pconnect()`;
- Soumission de la requête via la fonction `odbc_exec()`;
- Déconnexion grâce à la fonction `odbc_close()` (dans le cas d'une connexion non persistante).

Connexion

La connexion à la base de données s'opère grâce à la fonction `odbc_connect()`.

---

odbc_connect()

Établit une connexion (non persistante) avec le serveur de bases de données.

Syntaxe

```
resource odbc_connect(string $baseDonnees , string
$utilisateur, string $motDePasse [, int $typeCurseur])
```

`$baseDonnees`	Nom de la base de données (ou nom de la liaison ODBC).
`$utilisateur`	Nom de l'utilisateur.
`$motDePasse`	Mot de passe de l'utilisateur.
`$typeCurseur`	Sélectionne un des modes suivants : `SQL_CUR_USE_IF_NEEDED` (non supporté par DB2). `SQL_CUR_USE_ODBC` (non supporté par DB2). `SQL_CUR_USE_DRIVER`. `SQL_CUR_DEFAULT` (non supporté par DB2).
`retour`	Identifiant de connexion au serveur de bases de données, ou `FALSE` en cas d'échec.

Il est également possible d'utiliser une connexion persistante.

odbc_pconnect()

Établit une connexion persistante avec le serveur de bases de données.

Syntaxe	`resource odbc_pconnect(string $baseDonnees , string $utilisateur, string $motDePasse [, int $typeCurseur])`
`$baseDonnees`	Nom de la base de données (ou nom de la liaison ODBC).
`$utilisateur`	Nom de l'utilisateur.
`$motDePasse`	Mot de passe de l'utilisateur.
`$typeCurseur`	Sélectionne un des modes suivants : `SQL_CUR_USE_IF_NEEDED` (non supporté par DB2). `SQL_CUR_USE_ODBC` (non supporté par DB2). `SQL_CUR_USE_DRIVER`. `SQL_CUR_DEFAULT` (non supporté par DB2).
`retour`	Identifiant de connexion au serveur de bases de données, ou `FALSE` en cas d'échec.

Exécution de la requête SQL

L'exécution d'une requête SQL s'opère par un simple appel à `odbc_exec()`.

odbc_exec()

Exécute une requête SQL.

Syntaxe	`resource odbc_exec(resource $idConnexion, string $requete)`
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `odbc_connect()` ou `odbc_pconnect()`.

`$requete`	Requête SQL.
`retour`	Identifiant de résultat de requête SQL, ou `FALSE` en cas d'échec (identifiant de base de données ou requête SQL non valide).

Notez également l'existence de `odbc_do()` qui est un alias de `odbc_exec()`.

Nous verrons par la suite qu'il est également possible d'utiliser des requêtes préparées (contenant des paramètres qui pourront être modifiés juste avant que celles-ci ne soient exécutées) et de désactiver le mode `auto commit` (validation automatique) pour avoir accès aux opérations de `commit` (validation) et `rollback` (annulation).

Dans le cas d'une requête retournant un résultat (typiquement une requête `SELECT`), il faudra également analyser le résultat, mais nous verrons cela un peu plus loin.

Déconnexion

Pour vous déconnecter – opération théoriquement facultative mais toutefois vivement conseillée –, vous disposez de la fonction `odbc_close()`.

odbc_close()

Met fin à la connexion à la base de données et libère les ressources associées.

Syntaxe	`void odbc_close(resource $idConnexion)`
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `odbc_connect()` ou `odbc_pconnect()`.

Il existe également une fonction mettant fin à toutes les connexions ouvertes durant l'exécution du script.

odbc_close_all()

Ferme toutes les connexions et libère les ressources associées.

Syntaxe	`void odbc_close_all(void)`

Premier exemple

Nous voilà donc à même de construire notre premier script utilisant une base de données.

Vérifiez les paramètres
Prenez garde ! avant d'exécuter ce script, assurez-vous que les paramètres prédéfinis ne risquent pas de conduire à la suppression des données d'une de vos bases qui, par coïncidence, existerait déjà.

Comme cela est fortement conseillé, nous avons, ici, isolé les paramètres de connexion dans un fichier aisément repérable.

Listing 10.45 : parametres_bd_inc.php (exemple avec MS Access)

```
<?php

    // Paramètres de connexion à la base de données
    // N'hésitez pas à les modifier selon vos besoins

    $typeServeur = "MSAccess"; // MSAccess, IBMDB2, ...
    $base        = "mabase";
    $utilisateur = "";
    $motDePasse  = "";

?>
```

Notre script principal sera donc :

Listing 10.46 : insert01.php

```
<?php

    // Paramètres du script
    require_once("parametres_bd_inc.php");
    $table = "tableexemple";

    // Inclusion du script contenant les fonctions
    require_once("insert01_bd_inc.php");

    // Connexion à la base de données
    $idConnexion = odbc_pconnect($base, $utilisateur, $motDePasse);
    if (!$idConnexion) {
        die("<b>Impossible de se connecter à la base de données</b>");
    }

    // Appel de la fonction principale
    if (EX_initialiseBD($idConnexion, $table)) {
        echo "Voilà, une nouvelle table avec quelques données, ".
             "vous pouvez le vérifier avec votre client de base ".
             " de données ou via les scripts suivants.";
    } else {
        echo "La création ou l'alimentation de la table à échouée.";
    }
```

```
    // Pas de déconnexion dans le cas d'une connexion persistante
    // odbc_close($idConnexion);
?>
```

et nécessitera :

Listing 10.47 : insert01_bd_inc.php

```
<?php

    /**
     * Fonction chargée de créer et d'alimenter une table
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la table
     **/
    function EX_initialiseBD($idConnexion, $table)
    {
        // Supprime la précédente table
        $requete = "DROP TABLE $table";
        @odbc_exec($idConnexion, $requete);

        // Crée la table
        $requete = "CREATE TABLE $table (filmId INTEGER, film VARCHAR(64))";
        if (!odbc_exec($idConnexion, $requete)) return FALSE;

        // Ajoute quelques données
        $requete = "INSERT INTO $table VALUES (1, 'Forrest Gump')";
        if (!odbc_exec($idConnexion, $requete)) return FALSE;
        $requete = "INSERT INTO $table VALUES (2, 'Matrix')";
        if (!odbc_exec($idConnexion, $requete)) return FALSE;
        $requete = "INSERT INTO $table VALUES (3, 'La cité de la peur')";
        if (!odbc_exec($idConnexion, $requete)) return FALSE;

        return TRUE;
    }
?>
```

Champ auto-incrémenté

Il aurait été préférable de définir filmId comme étant un champ auto-incrémenté, plutôt que de le fixer manuellement. Mais, comme nous le verrons dans un prochain exemple, la syntaxe varie d'une base de données à l'autre.

Commit/rollback

Par défaut, les requêtes sont automatiquement validées (mode `auto commit`). Il est toutefois possible de modifier ce comportement par défaut en utilisant `odbc_autoCommit()`.

odbc_autoCommit()

Active, désactive ou retourne le mode `auto commit`.

Syntaxe	`mixed odbc_autoCommit(resource $idConnexion [, boolean $autoCommit])`
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `odbc_connect()` ou `odbc_pconnect()`.
`$autoCommit`	Indiquer `TRUE` pour activer le mode `auto commit`, `FALSE` pour désactiver le mode `auto commit`, ne pas renseigner pour récupérer le mode actuel.
`retour`	En mode lecture, `1` (mais pas strictement `TRUE`) si le mode `auto commit` est activé, `0` (mais pas strictement `FALSE`) sinon. En mode sélection, `TRUE` en cas de succès, `FALSE` sinon.

Si vous avez opté pour un mode de validation non automatique, vous pourrez valider ou annuler vos transactions avec les fonctions suivantes :

odbc_commit()

Valide les transactions.

Syntaxe	`boolean odbc_commit(resource $idConnexion)`
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `odbc_connect()` ou `odbc_pconnect()`.
`retour`	`TRUE` en cas de succès, ou rien (mais pas `FALSE`) sinon (c'est essentiellement le cas pour un identifiant non valide).

odbc_rollback()

Annule les transactions.

Syntaxe	`boolean odbc_rollback(resource $idConnexion)`
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `odbc_connect()` ou `odbc_pconnect()`.

`retour`	`TRUE` en cas de succès, ou rien (mais pas `FALSE`) sinon (c'est essentiellement le cas pour un identifiant non valide).

Lecture des enregistrements

Dans le cas d'une requête retournant une liste de données (typiquement un `SELECT`), il convient de récupérer l'identifiant de requête et de l'utiliser pour lire, en boucle, ligne après ligne, chaque enregistrement. Il existe diverses fonctions, chacune permettant de récupérer les champs des enregistrements sous différentes formes.

Les informations peuvent ainsi être retournées :

- Champ par champ ;
- Enregistrement par enregistrement (sous la forme d'un tableau indexé).

Lecture champ par champ

Pour une lecture champ par champ, vous pouvez passer d'un enregistrement au suivant par l'appel à `odbc_fetch_row()`, et accéder à l'un quelconque des champs par appel à `odbc_result()`.

odbc_fetch_row()

Passe à l'enregistrement suivant ou accède à n'importe quel enregistrement.

Syntaxe	`boolean odbc_fetch_row(resource $idResultat [, int $idxEnregistrement])`
`$idResultat`	Identifiant de résultat tel que retourné par `odbc_exec()` ou `odbc_execute()`.
`$idxEnregistrement`	Index de l'enregistrement à lire (le premier enregistrement ayant l'index 1). Par défaut, il s'agit de l'enregistrement suivant.
`retour`	`TRUE` en cas de succès, `FALSE` sinon (plus d'enregistrement à lire).

odbc_result()

Retourne la valeur d'un champ de l'enregistrement courant.

Syntaxe	`mixed odbc_result(resource $idResultat, mixed $champ)`
`$idResultat`	Identifiant de résultat tel que retourné par `odbc_exec()` ou `odbc_execute()`.
`$champ`	Au choix, soit le nom du champ (insensible à la casse), soit l'index du champ dans la requête (le premier champ ayant l'index 1).

retour	La valeur du champ, ou FALSE en cas d'erreur, mais également dans le cas où le champ est NULL ! En cas d'erreur, un message d'erreur est levé (voir odbc_error() et odbc_errormsg()), mais pas dans le cas d'un champ NULL.

Listing 10.48 : select_cc_bd_inc.php

```
<?php

    /**
     * Fonction listant le contenu d'une table
     * contenant 2 champs (filmId et film)
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la table
     **/
    function EX_listeContenu($idConnexion, $table)
    {
        // Requete
        $requete = "SELECT * FROM $table";
        $idResultat = odbc_exec($idConnexion, $requete);
        if (!$idResultat) return FALSE;

        // Boucle de lecture (et d'affichage) des enregistrements
        while (odbc_fetch_row($idResultat)) {
            echo "FilmId=".odbc_result($idResultat, "filmid")." ".
                 "Film =" .odbc_result($idResultat, "film")."<br />";
        }
        return TRUE;
     }
?>
```

Différence entre odbc_fetch_row() et mysql_fetch_row()

Notez que les fonctions `odbc_fetch_row()` *et* `mysql_fetch_row()` *ne jouent pas exactement le même rôle.* `mysql_fetch_row()` *regroupe les rôles de* `odbc_fetch_row()` *et* `odbc_result()`.

Lecture enregistrement par enregistrement

La fonction `odbc_fetch_into()` peut, dans certains cas, être plus pratique, puisqu'elle retourne l'ensemble des champs de l'enregistrement courant sous la forme d'un tableau. L'inconvénient de cette fonction, c'est que l'on ne peut accéder aux champs que par des index (et pas par des noms).

odbc_fetch_into()

Retourne l'enregistrement courant sous la forme d'un tableau indexé.

Syntaxe	`int odbc_fetch_into(resource $idResultat, array &$enregistrement [, int $idxEnregistrement])`
`$idResultat`	Identifiant de résultat tel que retourné par `odbc_exec()` ou `odbc_execute()`.
`$enregistrement`	Référence sur un tableau dans lequel seront copiées les données de l'enregistrement. Ce tableau est un tableau indexé dans lequel les valeurs apparaissent dans l'ordre des champs de la requête (le premier champ ayant l'index 0).
`$idxEnregistrement`	Index de l'enregistrement à lire (le premier enregistrement ayant l'index 1). Par défaut, il s'agit de l'enregistrement suivant.
`retour`	Le nombre de champs en cas de succès, `FALSE` sinon (plus d'enregistrement à lire).

Listing 10.49 : select_eei_bd_inc.php

```
<?php

    /**
     * Fonction listant le contenu d'une table
     * contenant 2 champs (filmId et film)
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la table
     **/
    function EX_listeContenu($idConnexion, $table)
    {
        // Requete
        $requete = "SELECT * FROM $table";
        $idResultat = odbc_exec($idConnexion, $requete);
        if (!$idResultat) return FALSE;

        // Boucle de lecture (et d'affichage) des enregistrements
        while (odbc_fetch_into($idResultat, $enreg)) {
            echo "FilmId=".$enreg[0]." ".
                 "Film =" .$enreg[1]."<br />";
        }
        return TRUE;
     }
?>
```

Libération des ressources

Si, au cours d'un script, vous faites appel à de nombreuses requêtes retournant de nombreux enregistrements, alors, n'hésitez pas, une fois le résultat de la requête exploité, à libérer les ressources associées avec la fonction `odbc_free_result()`.

odbc_free_result()

Libère les ressources allouées pour un résultat de requête.

Syntaxe	`boolean odbc_free_result(resource $idResultat)`
`$idResultat`	Identifiant de requête tel que retourné par `mysql_query()`.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

Nombre d'enregistrements

Nombre d'enregistrements retournés

Il existe trois façons de déterminer le nombre d'enregistrements retournés par une requête.

Si vous souhaitez uniquement connaître le nombre d'enregistrements, mais ne souhaitez pas immédiatement lire ces enregistrements, alors il vous suffit de faire une requête `COUNT()`, comme suit :

Listing 10.50 : count_bd_inc.php

```
<?php

    /**
     * Fonction affichant le nombre d'enregistrements
     * dans une table
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la tabme
     **/
    function EX_compte($idConnexion, $table)
    {
        // Requête
        $requete = "SELECT COUNT(*) FROM $table";
        $idResultat = odbc_exec($idConnexion, $requete);

        // Lecture du résultat
        if (odbc_fetch_into($idResultat, $enreg)) {
            echo "Il y a ".$enreg[0]." enregistrements dans la table.<br />";
            return TRUE;
        } else {
            echo "Etes vous sûr que la table $table existe ?<br />";
```

```
            return FALSE;
}
    }
?>
```


Si vous souhaitez connaître le nombre d'enregistrements, mais que vous souhaitez lire les enregistrements sans pour autant avoir besoin au préalable d'en connaître le nombre, il suffit de les compter au fur et à mesure de leur lecture.

Listing 10.51 : count_select_bd_inc.php

```
<?php

    /**
     * Fonction listant le contenu d'une table
     * contenant 2 champs (filmId et film)
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la table
     **/
    function EX_listeContenu($idConnexion, $table)
    {
        // Requete
        $requete = "SELECT * FROM $table";
        $idResultat = odbc_exec($idConnexion, $requete);
        if (!$idResultat) return FALSE;

        // Boucle de lecture (et de comptage) des enregistrements
        $nb = 0;
        while (odbc_fetch_into($idResultat, $enreg)) {
            // Vous pouvez manipuler $enreg à votre guise
            //echo "FilmId=".$enreg[0]." ".
            //     "Film =" .$enreg[1]."<br />";
            $nb++;
        }
        echo "Il y a $nb enregistrements dans la table.<br />";

        return TRUE;
     }
?>
```

Et, pour finir, ce que vous attendez tous : si vous souhaitez connaître le nombre d'enregistrements avant de les lire, vous pouvez faire appel à la fonction `odbc_num_rows()`.

odbc_num_rows()

Retourne le nombre d'enregistrements retournés ou modifiés par la requête SQL.

Syntaxe	`int odbc_num_rows(resource $idResultat)`
`$idResultat`	Identifiant de résultat tel que retourné par `odbc_exec()` (ne fonctionne pas avec `odbc_execute()`).
`retour`	Nombre d'enregistrements, ou rien (mais pas `FALSE`) en cas d'erreur.

Listing 10.52 : count_num_rows_bd_inc.php

```
<?php

    /**
     * Fonction listant le contenu d'une table
     * contenant 2 champs (filmId et film)
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la table
     **/
    function EX_listeContenu($idConnexion, $table)
    {
        // Requete
        $requete = "SELECT * FROM $table";
        $idResultat = odbc_exec($idConnexion, $requete);
        if (!$idResultat) return FALSE;

        echo "Il y a ".odbc_num_rows($idResultat)." enregistrements ".
             "dans la table<br />";

        return TRUE;
     }
?>
```

Sachant qu'une requête `SELECT *` est plus longue qu'une requête `SELECT COUNT(*)`, privilégiez la première méthode si vous n'avez que faire du contenu des enregistrements.

Nombre d'enregistrements modifiés

Pour déterminer le nombre d'enregistrements modifiés, vous pouvez appeler la fonction `odbc_num_rows()` présentée précédemment.

Mise à profit des requêtes préparées

Certains serveurs de bases de données permettent l'analyse, la compilation et le stockage des requêtes avant utilisation. Ceci permet d'exécuter une série de requêtes similaires, sans avoir à

renouveler à chaque fois les opérations d'analyse et de compilation (mais seulement en changeant certaines valeurs).

Pour cela, il suffit de préparer une requête dans laquelle les éléments variables sont remplacés par des points d'interrogation (ex. : `INSERT INTO matable (film) VALUES (?)`). Il suffira ensuite de préciser ces valeurs au moment de son exécution.

La préparation de la requête se fait via la fonction `odbc_prepare()`.

odbc_prepare()

Prépare une requête SQL.

Syntaxe	`resource odbc_prepare(resource $idConnexion, string $requete)`
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `odbc_connect()` ou `odbc_pconnect()`.
`$requete`	Requête SQL.
`retour`	Identifiant de requête en cas de succès (peut également être le cas pour des requêtes SQL non valides), ou rien (et non `FALSE`) en cas d'échec (identifiant de connexion non valide).

odbc_execute()

Exécute une requête préparée.

Syntaxe	`boolean odbc_execute(resource $idRequete [, array $tableauValeurs])`
`$idRequete`	Identifiant de requête préparée tel que retourné par `odbc_prepare()`.
`$tableauValeurs`	Tableau indexé contenant les différentes valeurs des éléments variables de la requête préparée (le premier paramètre ayant l'index 0).
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

DB2 d'IBM

Nous avons rencontré des problèmes avec l'utilisation des requêtes préparées sous DB2. En effet, nous n'avons pu utiliser qu'une seule requête préparée par script (ce qui nous condamnait à utiliser `odbc_exec()` pour les requêtes suivantes).

Gestion des erreurs

Jusque-là, en cas d'erreur, nous nous sommes contentés d'afficher un message générique. Il est cependant possible de déterminer plus précisément l'origine de l'erreur.

C'est possible notamment en récupérant un code d'erreur via la fonction `odbc_error()`.

odbc_error()

Retourne le dernier code d'erreur rencontré ou associé à la connexion.

Syntaxe	`string odbc_error([resource $idConnexion])`
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `odbc_connect()` ou `odbc_pconnect()`.
`retour`	Si l'identifiant de connexion est précisé : dernier code d'erreur (sur 5 chiffres) rencontré, ou chaîne vide en cas de succès de la dernière opération. (Il a été constaté que la chaîne n'est pas toujours exactement vide, mais peut contenir un caractère de contrôle). Si l'identifiant de connexion n'est pas précisé : dernier code d'erreur (sur 5 chiffres), même si la dernière opération s'est effectuée avec succès.

odbc_errormsg()

Retourne le dernier message d'erreur rencontré ou associé à la connexion.

Syntaxe	`string odbc_errormsg([resource $idConnexion])`
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `odbc_connect()` ou `odbc_pconnect()`.
`retour`	Si l'identifiant de connexion est précisé : dernier message d'erreur rencontré, ou chaîne vide en cas de succès de la dernière opération. (Il a été constaté que la chaîne n'est pas toujours exactement vide, mais peut contenir un caractère de contrôle). Si l'identifiant de connexion n'est pas précisé : dernier message d'erreur, même si la dernière opération s'est effectuée avec succès.

Exemples d'applications

Compteur de clics

Une application courante d'utilisation des bases de données et simple à mettre en œuvre consiste à créer un compteur de clics.

En effet, peut-être souhaitez-vous, sur votre site, proposer un certain nombre de liens vers des sites Internet (annuaire web) ou vers des fichiers (bibliothèques de scripts). Peut-être que certains (ou la totalité) de ces liens sont proposés par un partenaire. Bref, tout cela vous incite à savoir quelles sont les pages les plus fréquemment consultées (pour connaître les centres d'intérêt des visiteurs) et combien de visiteurs vous avez redirigés vers votre sponsor.

La méthode la plus classique consiste à stocker, en base de données, l'ensemble des liens ainsi proposés, et à remplacer les traditionnels liens de la forme **http://www.domaine.com** par un lien vers un script chargé d'incrémenter le compteur correspondant au site indiqué, et de rediriger le client vers le site grâce à la fonction `header()`.

Vous pouvez vous reporter à l'annexe "Les en-têtes" pour plus de renseignements sur la redirection.

Pour notre exemple, nous utiliserons une table appelée *url* contenant trois champs :

- *urlid*, identifiant de l'URL (champ auto-incrémenté) ;
- *url*, l'URL proprement dite ;
- *nbclic*, le nombre de clics.

Le script de redirection (appelé ici *compteurclic_redirection.php*) devra alors accepter un paramètre (`urlid`) et devra, à partir de ce paramètre, déterminer quelle est la valeur du champ *url*, incrémenter *nbclic* pour cette URL, et rediriger vers *url*.

Les appels à ce script auront alors la forme **http://localhost/compteurclic-redirection.php?urlid=3** (pour accéder à l'URL ayant l'identifiant 3). À supposer que l'URL 3 corresponde au site **http://www.sqlfacile.com** cela signifie que le lien `<a href="http://www.sqlfacile.com">` devra être remplacé par `<a href="compteurclic_redirection.php?urlid=3>` si l'on souhaite en compter le nombre de clics.

Concrètement, cette table pourra être créée et alimentée avec la fonction `CC_initialiseBD()` du script suivant :

Listing 10.53 : compteurclic_bd_inc.php (début)

```
<?php
    //-----------------------------------
    // Charge les paramètres de connexion
    // à la base de données.
    //-----------------------------------
    require_once("parametres_bd_inc.php");
```

```
/**
 * Fonction de connexion à une base de données
 * s'appuie sur les paramètres fournis
 * dans parametres_bd_inc.php.
 *
 * @return resource Identifiant de connexion
 */
function CC_connexion()
{
    global $base, $utilisateur, $motDePasse;

    return @odbc_pconnect($base, $utilisateur, $motDePasse);
}

/**
 * Fonction de deconnexion.
 * (Ne fait rien sachant que la fonction
 * de connexion établit une connexion persistante)
 */
function CC_deconnexion()
{
    return TRUE;
}

/**
 * Fonction chargé de créer et d'alimenter
 * la table "compteur de clics"
 **/
function CC_initialiseBD($idConnexion, $table)
{
    global $typeServeur;

    switch ($typeServeur) {
        case "MSAccess" :
            $compteur = "COUNTER";
            $default  = "";
            break;
        case "IBMDB2" :
            $compteur = "INTEGER GENERATED BY DEFAULT AS IDENTITY";
            $default  = "DEFAULT 0";
            break;
        default :
            $compteur = "INTEGER AUTO_INCREMENT";
            $default  = "DEFAULT 0";
    }

    // Création de la table
    $requete = "DROP TABLE $table";
    @odbc_exec($idConnexion, $requete);

    $requete = "CREATE TABLE $table (urlId $compteur,".
                                "url VARCHAR(128) NOT NULL,".
```

```
                                "nbclic INTEGER $default,".
                                "UNIQUE(url))";
        if (!odbc_exec($idConnexion, $requete)) return FALSE;

        // Alimentation de la table
        $requete = "INSERT INTO $table (url)".
                        " VALUES ('http://www.php.net')";
        if (!odbc_exec($idConnexion, $requete)) return FALSE;

        $requete = "INSERT INTO $table (url)".
                        " VALUES ('http://www.phpfacile.com')";
        if (!odbc_exec($idConnexion, $requete)) return FALSE;

        $requete = "INSERT INTO $table (url)".
                        " VALUES ('http://www.sqlfacile.com')";
        if (!odbc_exec($idConnexion, $requete)) return FALSE;

        $requete = "INSERT INTO $table (url)".
                        " VALUES ('http://www.xmlfacile.com')";
        if (!odbc_exec($idConnexion, $requete)) return FALSE;

        $requete = "INSERT INTO $table (url)".
                        " VALUES ('http://www.ootoogo.com')";
        if (!odbc_exec($idConnexion, $requete)) return FALSE;

        if ($typeServeur == "MSAccess") {
            // Un "patch" pour MSAccess qui ne supporte pas DEFAULT
            $requete = "UPDATE $table SET nbclic=0";
            if (!odbc_exec($idConnexion, $requete)) return FALSE;
        }

        return TRUE;
    }
?>
```

REMARQUE

Spécificité des bases de données

Même si les fonctions ODBC permettent d'accéder à de nombreuses bases de données, il n'en reste pas moins que le langage SQL supporté diffère de l'une à l'autre. C'est ce que nous pouvons constater ici en ce qui concerne la création d'un champ auto-incrémenté ou les valeurs par défaut. `DEFAULT` *n'étant pas supporté par MS Access 97, il convient de fixer manuellement les valeurs par défaut.*

La lecture de la liste des liens disponibles en base de données pourra se faire via la fonction `CC_recupereLiens()` du script suivant :

Listing 10.54 : compteurclic_bd_inc.php (milieu)

```
<?php
    /**
     * Fonction retournant les informations de liens
```

```
     * sous forme d'un tableau associatif possedants
     * les clés
     * - "lien" associé au tableau des liens hypertextes
     * - "nbclic" associé au tableau des nombres de clics
     **/
    function CC_recupereLiens($idConnexion, $table)
    {
        // Nom du script chargé du comptage et de la redirection
        $script = "compteurclic_redirection.php";

        // Requête SELECT
        $requete = "SELECT * FROM $table";
        $idResultat = odbc_exec($idConnexion, $requete);

        // Récupération des enregistrements les uns après les autres
        while (odbc_fetch_row($idResultat)) {
            $liens["lien"][] = "<a href=\"$script?urlid=".
                               odbc_result($idResultat, "urlid")."\">".
                               odbc_result($idResultat, "url")."</a>";
            $liens["nbclic"][] = odbc_result($idResultat, "nbclic");
        }

        return $liens;
    }
?>
```

L'affichage des liens consiste uniquement à mettre en page le contenu du tableau ainsi récupéré (voir la fonction du script *compteurclic_html_inc.php*).

Comme cela a été précisé, le lien `<a href="http://www.sqlfacile.com">http://www.sqlfacile.com</a>` a été remplacé par `<a href="compteurclic_redirection.php?urlid=3">http://www.sqlfacile.com </a>`. La redirection vers le site **www.sqlfacile.com** est donc à la charge du script *compteurclic_redirection.php*.

Listing 10.55 : compteurclic_redirection.php

```
<?php

    // Paramètres du script
    require_once("compteurclic_bd_inc.php");
    $table = "compteurclic";

    // Connexion à la base de données
    $idConnexion = @CC_connexion();
    if (!$idConnexion) {
        die("<b>Impossible de se connecter à la base de données</b>");
    }

    // Récupération de l'url et incrémentation du compteur
    // pour l'url d'indentifiant passé en paramètre de ce script
    // par la méthode GET
    $url = @CC_recupereUrl($idConnexion, $_GET["urlid"]);
```

```
    // Deconnexion
    @CC_deconnexion();

    // C'est l'heure de la redirection
    if ($url) {
        header("Location: $url");
    } else {
        echo "Désolé, nous ne pouvons vous proposer ce lien";
    }
?>
```

Ce script appelle principalement la fonction `CC_recupereUrl()` suivante :

Listing 10.56 : compteurclic_bd_inc.php (fin)

```
<?php
    /**
     * Fonction récupérant une url à partir de son identifiant
     * et incrémentant le compteur de clics
     **/
    function CC_recupereUrl($idConnexion, $urlid)
    {
        global $table;

        // Récupère l'url
        $requete = "SELECT * FROM $table WHERE urlid=$urlid";
        $idResultat = odbc_exec($idConnexion, $requete);

        if (odbc_fetch_row($idResultat)) {
            $url = odbc_result($idResultat, "url");

            // Incrémente le compteur
            $requete = "UPDATE $table SET nbclic=nbclic+1".
                                    " WHERE urlid=$urlid";
            odbc_exec($idConnexion, $requete);
        } else {
            $url = FALSE;
        }
        return $url;
    }
?>
```


REMARQUE

Utilisation du bouton retour

Si, après avoir suivi un lien, vous revenez sur la page compteurclic.php en utilisant le bouton retour (back), le contenu de la page ne sera pas réactualisé. Par conséquent, le nombre de visites affiché ne sera pas correct. N'oubliez donc pas, dans ce cas, de rafraîchir (bouton actualiser) la page.

SuperTheque

L'essentiel du code de l'application a été présenté en introduction de ce chapitre. Il ne restait plus qu'a connaître les fonctions SQLite pour implémenter l'interface `RessourceInterface` c'est désormais chose faite:

Listing 10.57 : RessourceODBC_class.php

```
<?php
include_once("RessourceInterface_class.php");
include_once(dirname(__FILE__)."/../config/odbc_cfg.php");
/**
 * RessourceODBC_class.php
 * Classe d'acces a une base de donnees MS ACCESS ou IBM DB2 via ODBC
 * Cette classe doit implementer toutes les methodes de l'interface
 *   RessourceInterface.
 * Compatibilite: PHP 5
 */
class Ressource implements RessourceInterface
{
    var $idConnexion;

    public function connexion()
    {
        global $odbc_serveur, $odbc_utilisateur, $odbc_motDePasse;
        global $odbc_base;

        if (!isset($this->idConnexion)) {
            $idConnexion = odbc_pconnect($odbc_base,
                                         $odbc_utilisateur,
                                         $odbc_motDePasse);
            if (!$idConnexion) return FALSE;
            $this->idConnexion = $idConnexion;
        }
        return TRUE;
    }

    public function deconnexion()
    {
        // Rien a faire, il s'agit d'une connexion persistante
    }

    public function reset()
    {
        global $odbc_typeServeur;

        $requete = "DROP TABLE types";
        $idResultat = @odbc_exec($this->idConnexion, $requete);

        switch ($odbc_typeServeur) {
            case "MSAccess" :
               $compteur = "COUNTER";
```

```
            break;
        case "IBMDB2" :
            $compteur = "INTEGER GENERATED BY DEFAULT AS IDENTITY";
            break;
        default:
            $compteur = "INTEGER AUTO_INCREMENT";
     }

    $requete = "CREATE TABLE types (".
                    "id $compteur,".
                    "type VARCHAR(255))";
    $idResultat = odbc_exec($this->idConnexion, $requete);
    if (!$idResultat) return FALSE;

    $types = array("Album", "Film", "Livre", "Musique");
    foreach ($types as $type) {
        $requete = "INSERT INTO types (type) VALUES ('$type')";
        $idResultat = odbc_exec($this->idConnexion, $requete);
        if (!$idResultat) return FALSE;
    }

    $requete = "DROP TABLE articles";
    $idResultat = @odbc_exec($this->idConnexion, $requete);

    $requete = "CREATE TABLE articles (".
                 "id $compteur,".
                 "albumId INTEGER,".
                 "titre VARCHAR(255),".
                 "typeId INTEGER,".
                 "commentaire VARCHAR(255) NULL)";
    $idResultat = odbc_exec($this->idConnexion, $requete);
    if (!$idResultat) return FALSE;
}

public function addArticle($albumId,
                           $titre,
                           $typeId,
                           $commentaire)
{
    $requeteDebut = "INSERT INTO articles (albumId, titre, typeId";
    $requeteFin = ") VALUES ($albumId,".
                            "'".addSlashes($titre)."',".
                            "$typeId";
    if ($commentaire != "") {
         $requeteDebut .= ",commentaire";
         $requeteFin .= ",'".addSlashes($commentaire)."'";
    }
    $requete = $requeteDebut . $requeteFin . ")";
    $idResultat = odbc_exec($this->idConnexion, $requete);
    if (!$idResultat) return FALSE;
}
```

```
public function getArticle($articleId)
{
    $requete = "SELECT * FROM articles WHERE id=$articleId";
    $idResultat = odbc_exec($this->idConnexion, $requete);
    if (!$idResultat) return FALSE;

    $article = NULL;
    if ($enreg = odbc_fetch_array($idResultat)) {
        $article = $this->enreg2Article($enreg);
    }
    return $article;
}

public function getArticles(Filtre $filtre, Plage $plage, Tri $tri)
{
    $requete = "SELECT * FROM articles";
    if ($filtre->getAlbumId() != -1) {
        $conditionSQL = "albumId=".$filtre->getAlbumId();
    }
    if ($filtre->getTitre() != "") {
        $conditionSQL = $conditionSQL.
                        " AND titre LIKE '".
                        addSlashes($filtre->getTitre())."'";
    }
    if ($filtre->getTypeId() != -1) {
        $conditionSQL = $conditionSQL.
                        " AND typeId=".$filtre->getTypeId();
    }

    if ($tri->getSens() == -1) {
      $triSQL = "ORDER BY ".$tri->getChamp()." DESC";
    } else if ($tri->getSens() == 1) {
      $triSQL = "ORDER BY ".$tri->getChamp()." ASC";
    }

    $requete = $requete." WHERE ".$conditionSQL." ".
                                  $triSQL;
    $idResultat = odbc_exec($this->idConnexion, $requete);
    if (!$idResultat) return FALSE;

    $articles = NULL;
    // Nous devons sauter les premiers resultats
    if ($plage->getPremierArticle()>0) {
       odbc_fetch_row($idResultat,
                      $plage->getPremierArticle());
    }

    while (($enreg = odbc_fetch_array($idResultat))
           &&(count($articles)<$plage->getNbArticleMax())) {
        $articles[] = $this->enreg2Article($enreg);
    }
    return $articles;
```

```
}

public function getNbTotalArticles(Filtre $filtre)
{
    $requete = "SELECT COUNT(*) FROM articles";
    if ($filtre->getAlbumId() != -1) {
        $conditionSQL = "albumId=".$filtre->getAlbumId();
    }
    if ($filtre->getTitre() != "") {
        $conditionSQL = $conditionSQL.
                        " AND titre LIKE '".
                        addSlashes($filtre->getTitre())."'";
    }
    if ($filtre->getTypeId() != -1) {
        $conditionSQL = $conditionSQL.
                        " AND typeId=".$filtre->getTypeId();
    }

    $requete = $requete." WHERE ".$conditionSQL;
    $idResultat = odbc_exec($this->idConnexion, $requete);
    if (!$idResultat) return FALSE;
    if (odbc_fetch_into($idResultat, &$enreg)) {
        return $enreg[0];
    } else return FALSE;
}


public function getTypes()
{
    $requete = "SELECT * FROM types";
    $idResultat = odbc_exec($this->idConnexion, $requete);
    if (!$idResultat) return FALSE;

    $types = NULL;
    while ($enreg = odbc_fetch_array($idResultat)) {
        $types[$enreg["id"]] = $enreg["type"];
    }
    return $types;
}

public function getAlbumTypeId()
{
     return 1;
}


private function enreg2Article($enreg)
{
    $article = new Article();
    $article->setId($enreg["id"]);
    $article->setAlbumId($enreg["albumId"]);
    $article->setTitre($enreg["titre"]);
```

```
            $article->setTypeId($enreg["typeId"]);
            $article->setCommentaire($enreg["commentaire"]);
            return $article;
        }

}
?>
```


Comme vous le constatez, la principale difficulté n'est pas tant au niveau des fonctions disponibles qu'au niveau de la construction des requêtes SQL. La requête sera simple lorsqu'il ne s'agit que de récupérer un enregistrement identifié par la clé `Id` (comme c'est le cas avec `getArticle()`) elle sera bien plus complexe s'il s'agit des récupérer un ensemble d'enregistrements répondant à des critères précis et triés (comme c'est le cas avec `getArticles()`).

Requêtes avec des apostrophes

Lors de la construction de la requête, il faut bien garder à l'esprit que l'un des éléments de la requête (typiquement un champ de type texte saisi par l'utilisateur, comme ici le titre ou le commentaire) peut contenir une apostrophe qui pourrait être confondu avec le délimiteur de chaîne. Pensez donc bien à faire un appel à `addSlashes()` lorsque cela peut s'avérer nécessaire.

Affichage page par page

Certaines bases de données permettent de ne retourner qu'un sous-ensemble des résultats d'une requêtes SQL ; d'autres permettent seulement de retourner les N premiers résultats. En ce qui concerne MS Access et DB2, il semblerait qu'ils ne permettent pas ce genre d'optimisation. Nous sommes donc dans l'obligation de demander tous les résultats, et d'ignorer ceux qui ne nous intéressent pas.

Tri

Modifier l'ordre d'affichage des annonces n'est pas non plus bien sorcier. Le tri peut s'effectuer directement au niveau de la requête SQL, grâce à l'instruction `ORDER BY`.

Moteur de recherche (filtre)

Pour filtrer, il suffit d'utiliser l'instruction `WHERE`. Dans notre cas, nous souhaitons autoriser l'utilisation de jokers (comme %) afin, par exemple, de rechercher les titres commençant par une chaîne donnée. C'est pourquoi, nous ne ferons pas un test de type `titre='motcle'` mais `titre LIKE 'motcle'` (où `motcle` pourrait avoir la valeur "La 7eme compagnie%").

En savoir plus...

Il y a bien plus à apprendre d'une base de données...

… sur le résultat d'une requête

Il vous est ainsi possible de connaître le nombre de champs retournés par une requête (et par conséquent le nombre de champs d'une table dans le cas d'une requête `SELECT * FROM <nom_table>` sur une table non vide).

odbc_num_fiels()

Retourne le nombre de champs dans l'enregistrement retourné par une requête SQL.

Syntaxe	`int odbc_num_fields(resource $idResultat)`
`$idResultat`	Identifiant de résultat tel que retourné par `odbc_exec()` ou `odbc_execute()`.
`retour`	Nombre de champs, ou `FALSE` en cas d'erreur.

Mais aussi, le nom de ces champs :

odbc_field_name()

Retourne le nom d'un champ désigné par son indice dans le résultat de requête.

Syntaxe	`string odbc_field_name(resource $idResultat, int $idxChamp)`
`$idResultat`	Identifiant de résultat tel que retourné par `odbc_exec()` ou `odbc_execute()`.
`$idxChamp`	Index du champ (le premier champ porte l'index 1).
`retour`	Nom du champ, ou `FALSE` en cas d'erreur.

Ou, à l'inverse, leur index :

odbc_field_num()

Retourne l'index du champ désigné par son nom dans le résultat de requête.

Syntaxe	`int odbc_field_num(resource $idResultat, string $champ)`
`$idResultat`	Identifiant de résultat tel que retourné par `odbc_exec()` ou `odbc_execute()`.
`$champ`	Nom du champ.
`retour`	Index du champ (le premier champ porte l'index 1), ou `FALSE` en cas d'erreur.

Cependant, les champs ont d'autres informations à nous livrer que leur nom et leur index. Pour cela, vous disposez des fonctions :

odbc_field_type()

Retourne le type du champ désigné par son index dans le résultat de requête.

Syntaxe	`int odbc_field_type(resource $idResultat, int $idxChamp)`
`$idResultat`	Identifiant de résultat tel que retourné par `odbc_exec()` ou `odbc_execute()`.
`$idxChamp`	Index du champ (le premier champ porte l'index 1).
`retour`	Type du champ, ou `FALSE` en cas d'erreur.

odbc_field_len()

Retourne la longueur du champ désigné par son index dans le résultat de requête.

Syntaxe	`int odbc_field_len(resource $idResultat, int $idxChamp)`
`$idResultat`	Identifiant de résultat tel que retourné par `odbc_exec()` ou `odbc_execute()`.
`$idxChamp`	Index du champ (le premier champ porte l'index 1).
`retour`	Longueur du champ, ou `FALSE` en cas d'erreur.

Cette fonction possède un alias baptisé `odbc_field_precision()`

odbc_field_scale()

Retourne l'échelle (?) du champ désigné par son index dans le résultat de requête (0 souvent, 6 pour un TIMESTAMP DB2).

Syntaxe	`int odbc_field_scale(resource $idResultat, int $idxChamp)`
`$idResultat`	Identifiant de résultat tel que retourné par `odbc_exec()` ou `odbc_execute()`.
`$idxChamp`	Index du champ (le premier champ porte l'index 1).
`retour`	Echelle (?) du champ, ou `FALSE` en cas d'erreur.

... sur les champs d'une table

Il n'est cependant pas nécessaire de passer par une requête pour récupérer des informations sur une table. Vous disposez ainsi de :

odbc_columns()

Retourne un pointeur sur la liste des champs des tables d'une base.

Syntaxe `resource odbc_columns(resource $idConnexion)`

`$idConnexion` Identifiant de connexion à une base de données tel que retourné par `odbc_connect()` ou `odbc_pconnect()`.

`retour` Identifiant de requête (à analyser, par exemple, avec les fonctions `odbc_fetch*()`). Les champs sont alors :
TABLE_CAT.
TABLE_SCHEM.
TABLE_NAME, le nom de la table.
COLUMN_NAME, le nom du champ.
DATA_TYPE, le code du type du champ (voir tableau).
TYPE_NAME, le type du champ en toutes lettres.
COLUMN_SIZE, taille du champ (en octets).
BUFFER_LENGTH.
DECIMAL_DIGITS.
NUM_PREC_RADIX.
NULLABLE, `1` si le champ peut prendre la valeur `NULL`.
REMARKS, commentaire associé au champ.
COLUMN_DEF.
SQL_DATA_TYPE, le code SQL du type du champ.
SQL_DATETIME_SUB.
CHAR_OCTET_LENGTH.
ORDINAL_POSITION.
IS_NULLABLE, *YES* si le champ peut prendre la valeur `NULL`, *NO* sinon.
ORDINAL.

Tableau 10.15 : Les types ODBC rencontrés

Identifiant de Type	Access 97 (MS)	DB2 (IBM)
-400		DATALINK
-99		CLOB
-98		BLOB
-11	GUID	
-7	BIT	
-6	BYTE	
-5		BIGINT

Identifiant de Type	Access 97 (MS)	DB2 (IBM)
-4	LONGBINARY	LONG VARCHAR FOR BIT DATA
-3	VARBINARY	VARCHAR () FOR BIT DATA
-2	BINARY	CHARacter() FOR BIT DATA
-1	LONGCHAR	LONG VARCHAR
1	CHAR	CHARacter
2	CURRENCY	NUMeric
3		DECimal
4	INTEGER COUNTER	INTeger
5	SMALLINT	SMALLINT
6		FLOAT
7	REAL	REAL
8	DOUBLE	FLOAT
9		DATE
10		TIME
11	DATETIME	TIMESTAMP
12	VARCHAR	VARCHAR

... sur les tables d'une base

De même, vous pouvez aisément récupérer la liste des tables.

odbc_tables()

Retourne un pointeur sur la liste des tables d'une base.

Syntaxe `resource odbc_tables(resource $idConnexion)`

`$idConnexion` Identifiant de connexion à une base de données tel que retourné par `odbc_connect()` ou `odbc_pconnect()`.

`retour` Identifiant de requête (à analyser, par exemple, avec les fonctions `odbc_fetch*()`). Les champs sont alors :
TABLE_CAT.
TABLE_SCHEM.
TABLE_NAME, le nom de la table.
TABLE_TYPE, type de la table *TABLE*, *SYSTEM TABLE*, *VIEW*, etc.
REMARKS, commentaire associé à la table.

... sur les procédures stockées d'une base

De même, vous pouvez aisément récupérer la liste des procédures stockées.

odbc_procedures()

Retourne un pointeur sur la liste des procédures stockées d'une base.

Syntaxe	`resource odbc_procedures(resource $idConnexion)`
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `odbc_connect()` ou `odbc_pconnect()`.
`retour`	Identifiant de requête (à analyser, par exemple, avec les fonctions `odbc_fetch*()`). Les champs sont alors : *PROCEDURE_CAT*. *PROCEDURE_SCHEM*. *PROCEDURE_NAME*, le nom de la procédure stockée. *NUM_INPUT_PARAMS*, le nombre de paramètres d'entrée. *NUM_OUTPUT_PARAMS*, le nombre de paramètres de sortie. *REMARKS*, commentaire associé à la procédure stockée. *PROCEDURE_TYPE*, le type de la procédure stockée.

Tout comme les informations plus précises :

odbc_procedurecolumns()

Retourne un pointeur sur la liste des paramètres des procédures stockées d'une base.

Syntaxe	`resource odbc_procedurecolumns(resource $idConnexion)`
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `odbc_connect()` ou `odbc_pconnect()`.
`retour`	Identifiant de requête (à analyser, par exemple, avec les fonctions `odbc_fetch*()`). Les champs sont alors : *PROCEDURE_CAT*. *PROCEDURE_SCHEM*. *PROCEDURE_NAME*, le nom de la procédure stockée. *COLUMN_NAME*, le nom du paramètre. *COLUMN_TYPE*. *DATA_TYPE*, l'identifiant du type du paramètre (voir tableau). *TYPE_NAME*, le type du paramètre en toutes lettres. *COLUMN_SIZE*, taille du paramètre (en octets). *BUFFER_LENGTH*. *DECIMAL_DIGITS*. *NUM_PREC_RADIX*. *NULLABLE*, `1` si le champ peut prendre la valeur `NULL`.

REMARKS, commentaire associé au paramètre.
COLUMN_DEF.
SQL_DATA_TYPE, l'identifiant SQL du type du paramètre.
SQL_DATATYPE_SUB.
CHAR_OCTET_LENGTH.
ORDINAL_POSITION, position du paramètre dans l'appel de la procédure.
IS_NULLABLE, *YES* si le paramètre peut prendre la valeur `NULL`.

... sur le serveur

Il vous est possible d'accéder à la liste des types supportés par le serveur de bases de données en appelant la fonction `odbc_getTypeInfo()`.

odbc_getTypeInfo()

Lance une requête interrogeant la base de données sur la liste des types SQL supportés.

Syntaxe `resource odbc_getTypeInfo(resource $idConnexion [, int $idType])`

`$idConnexion` Identifiant de connexion à une base de données tel que retourné par `odbc_connect()` ou `odbc_pconnect()`.

`$idType` Identifiant de type sur lequel limiter la recherche (par défaut, ou si `$idType` vaut 0, la liste complète est retournée).

`retour` Identifiant de résultat de requête contenant les champs :
TYPE_NAME, type en toutes lettres.
DATA_TYPE, identifiant (numérique) du type.
COLUMN_SIZE.
LITERAL_PREFIX.
LITERAL_SUFFIX.
CREATE_PARAMS.
NULLABLE.
CASE_SENSITIVE.
SEARCHABLE.
UNSIGNED_ATTRIBUTE.
FIXED_PREC_SCALE.
AUTO_UNIQUE_VALUE.
SQL_DATA_TYPE.
SQL_DATETIME_SUB.
NUM_PREC_RADIX.
INTERVAL_PRECISION.

10.11. Oracle

Oracle est certainement la base de données la plus utilisée dans le monde professionnel ; il était donc important de décrire, ici, comment accéder à une telle base via PHP.

Installation

Le but de ce chapitre n'est pas de faire de vous un administrateur de base de données Oracle. Non. Nous nous contenterons de décrire une installation standard, sans prendre en compte les problèmes d'optimisation et de sécurité. Le but étant que vous puissiez installer PHP et Oracle sur des machines de test pour vous familiariser avec cet environnement avant de passer à un serveur de bases de données destiné à la production (et peut-être configuré par un expert Oracle).

De plus nous installerons le serveur Oracle sur la même machine que le serveur Web.

Pour nos tests, nous avons installé Oracle 10g sous un environnement Debian Sarge.

Dans tous les cas, vous êtes vivement invités à consulter la documentation officielle Oracle pour mener à bien l'installation de la base de données sur votre système.

Pré-requis

Pour commencer, vous devez vous procurer le CD-ROM d'installation d'Oracle ou en télécharger une version à l'adresse **http://otn.oracle.com/software/content.html**.

Pour Linux, la version 10.1.0.2 téléchargée se présente sous la forme d'un fichier *ship.db.cpio.gz*.

Attention, l'installation d'Oracle est très exigeante. Il vous faudra 512 Mo de RAM (en fait nous l'avons menée à bien avec à peu moins) ainsi qu'1 Go de swap.

En manque de swap ?
Si vous ne disposez pas suffisamment de swap vous pouvez aisément en créer (à condition de disposer de suffisamment d'espace disque). Pour cela enchaîner les commandes suivantes (vous pouvez la répéter pour créer plusieurs fichiers).

```
dd if=/dev/zero of=fichierswap bs=1k count=<taille en d'octets du fichier
✂ de swap>
chmod 600 fichierswap
mkswap fichierswap
swapon fichierswap
```

Préparation à l'installation du serveur de base de données

Vous êtes invité à créer, depuis le compte `root`, sur le serveur Oracle, un nouveau compte utilisateur `oracle` associé au groupe principal `oinstall` (groupe des fichiers installés par oracle) et au groupe `dba` (groupe des administrateurs oracle).

```
# groupadd oinstall
```

```
# groupadd dba
# useradd -g oinstall –G dba -s /bin/bash -d /home/oracle oracle
```

Pour vous éviter des soucis, vous pouvez créer dès maintenant le répertoire destiné à héberger la partie logicielle d'Oracle et celui pour les données.

```
# mkdir –p /usr/local/oracle10g
# mkdir –p /data/oradata
# chown –R oracle:oinstall /usr/local/oracle10g
# chmod –R 775 /usr/local/oracle10g /data/oradata
```

Il est maintenant temps de se logger en tant qu'utilisateur `oracle` :

```
# su - oracle
```

et de modifier l'environnement de ce compte utilisateur. Pour cela, modifiez le fichier *~oracle/.bashrc* en ajoutant les lignes suivantes :

```
export ORACLE_BASE=/usr/local/oracle10g
export ORACLE_HOME=$ORACLE_BASE/product/10.1.0/db_1
export ORACLE_SID=mabase
unset LANG
export PATH=$ORACLE_HOME/bin:$PATH
```

`ORACLE_BASE` est le répertoire où sera installé la partie logicielle d'Oracle incluant les scripts d'installation. Le moteur de la base de donnée étant quant à lui sous `ORACLE_HOME`.

`ORACLE_SID` est le nom de la base de données que l'on souhaite utiliser par défaut.

Pour hériter de ces modifications, tapez maintenant la commande :

```
$ source ~/.bashrc
```

Il est temps maintenant de décompresser l'archive précédemment téléchargée (par exemple, sous */usr/local/src/database/oracle10g*).

```
$ cd /usr/local/src/database/oracle10g
$ gunzip ship.db.cpio.gz
$ cpio –idmv < ship.db.cpio
```

Vous voilà désormais avec un répertoire */usr/local/src/database/oracle10g/Disk1* contenant, entre autres, un fichier *runInstaller*.


REMARQUE

Installation depuis un CD-ROM

Dans le cas d'une installation par CD-ROM, vous n'avez pas à décompresser d'archives, mais vous devez "monter" le CD. Cette opération se réalise généralement (mais cela dépend de l'environnement) par l'opération (depuis le compte `root`) :

```
# mount /mnt/cdrom
```

Dès lors, les opérations suivantes faisant référence à /usr/local/src/database/oracle10g/Disk1 doivent être comprises comme faisant référence à /mnt/cdrom.

Avant de lancer le script d'installation, assurez-vous que le compte `oracle` a le droit d'ouvrir une fenêtre X. Pour cela, pas la peine de faire de sentiments : ouvrez une nouvelle fenêtre (`xterm`) en tant que l'utilisateur initial de la session, et tapez la commande :

```
# xhost +
```

Vous pouvez alors tenter de lancer le script d'installation.

```
$ export DISPLAY=:0.0
$ cd /tmp/Disk1
$ ./runInstaller
```

Ceci affiche la fenêtre suivante :

Figure 10.19 : *Bienvenue*

Comment ? Cela ne fonctionne pas ? Mouais... Il fallait s'y attendre. Ce n'est sans doute pas bien grave ; consultez simplement la remarque qui suit. Si, en revanche, cela a fonctionné, vous avez de la chance (!), et vous pouvez passer à la suite.

REMARQUE

Distribution non supportée

Il se peut que l'exécution de la commande précédente conduise au message d'erreur suivant:

Checking operating system version: must be redhat-2.1, UnitedLinux-1.0 or redhat-3
Failed <<<<

Ceci est juste dû au fait qu'Oracle 10g ne supporte officiellement que les trois distributions citées dans le message, ce qui ne signifie toutefois pas qu'il ne fonctionne pas sous d'autres distributions.

REMARQUE

Pour éviter cela, deux solutions s'offrent à vous:
Vous pouvez lancer le script d'installation avec l'option permettant de ne pas faire le contrôle des pré-requis.

```
$ ./runInstaller -ignoreSysPrereqs
```

Ou bien, vous faire passer pour un autre (enfin... faire passer votre distribution par une autre)
Pour cela créez ou modifiez (après en avoir fait une sauvegarde) le fichier /etc/redhat-release afin qu'il contienne l'unique ligne:
Red Hat Enterprise Linux AS release 3 (Taroon)
et relancez runInstaller.

Une fois que vous aurez cliqué sur le bouton **Suivant**, vous verrez apparaître :

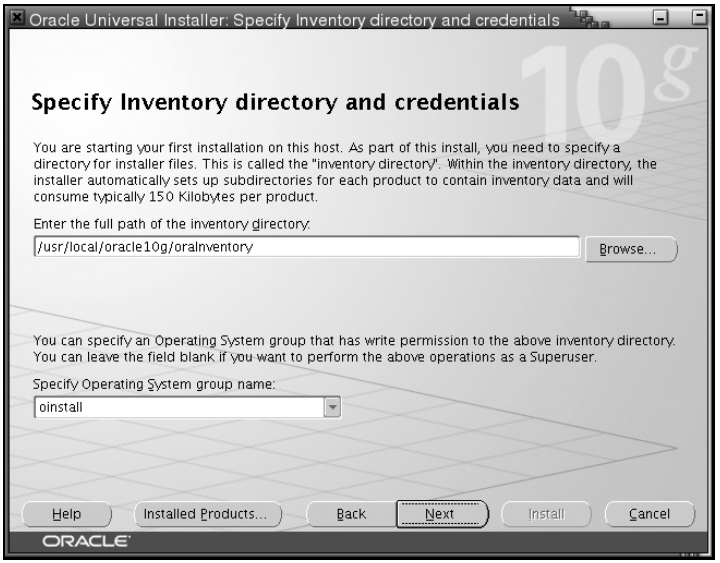


Figure 10.20 : *Emplacement des scripts d'installation*

Cette fenêtre de dialogue vous demande de confirmer le chemin du répertoire *oraInventory* où seront déposés les scripts d'installation (celui ci est basé sur la valeur définie par *ORACLE_HOME*) ainsi que le groupe auquel les fichiers ainsi créés doivent appartenir. Vous pouvez conserver les valeurs par défaut et valider.

Durant cette pré-installation, vous serez invités à exécuter un script depuis le compte utilisateur `root`.

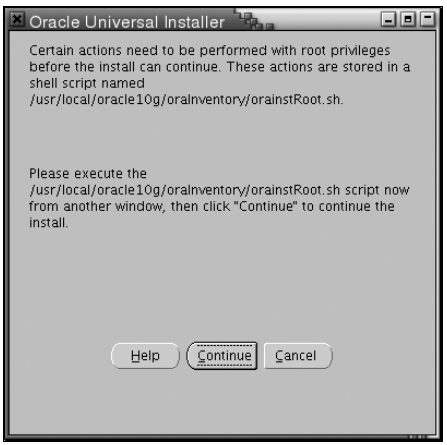


Figure 10.21 : *Pré-installation sous le compte root*

Pour cela, il suffit de suivre les instructions. Dans une fenêtre (`xterm`) ouvrez une session `root` et exécutez la commande

```
# /usr/local/oracle10g/oraInventory/orainstRoot.sh
```

Vous pouvez maintenant reprendre le cours normal de l'installation en cliquant sur **Continue**.

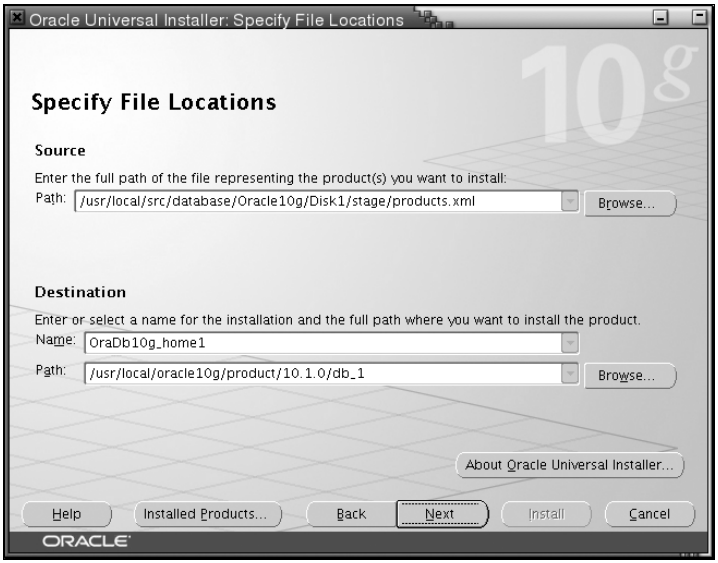


Figure 10.22 : *Emplacement des fichiers*

Il vous est maintenant demandé de préciser le chemin du répertoire contenant le logiciel d'installation ainsi que le chemin du répertoire destination. Là, encore, vous pouvez conserver les valeurs par défaut et valider. Le logiciel installe alors tous les scripts nécessaires à l'installation du serveur.

Vous êtes maintenant prêt à installer le serveur.

Installation du serveur Oracle

Une fois les étapes précédentes passées, vous arrivez à l'écran suivant :

Figure 10.23 : *Sélection du type d'installation*

Pour une première installation, nous nous contenterons d'une installation "standard".

Attention, si vous souhaitez installer le composant contenant les messages pour une langue autre que l'anglais, avant de cliquer sur **Next**, sélectionnez **Product Languages**.

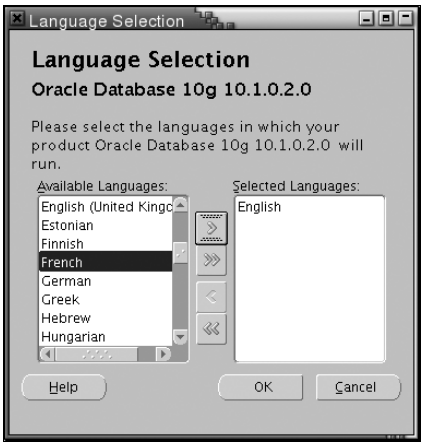


Figure 10.24 :
Sélection des langues

C'est tout simple, il suffit d'ajouter les langues désirées. Puis cliquez sur **Ok**. Et cliquez sur le bouton **Next** de la fenêtre principale (présentée précédemment).

Le script contrôle alors les différents paramètres de l'environnement...

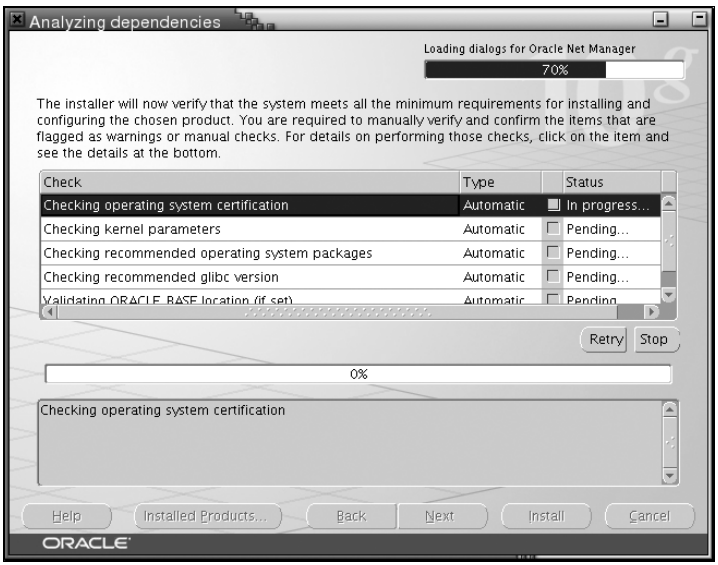


Figure 10.25 : *Analyse de l'environnement*

... et poursuit l'installation (apparemment y compris si des erreurs sont détectées)

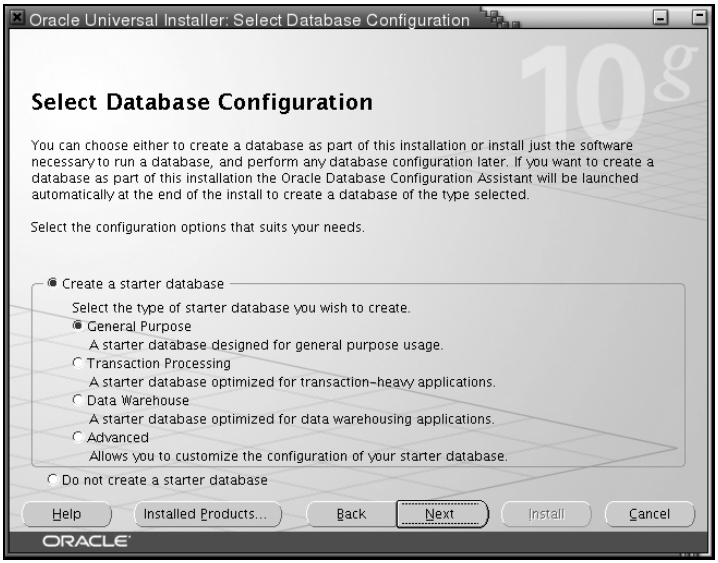


Figure 10.26 : *Création d'une base*

Vous pouvez choisir de n'installer que le serveur de la base de données mais autant en profiter pour créer également la base de données. Pour cela, conservez les valeurs par défaut, à savoir "Create a starter database" de type "General Purpose", puis cliquez sur **Next**.

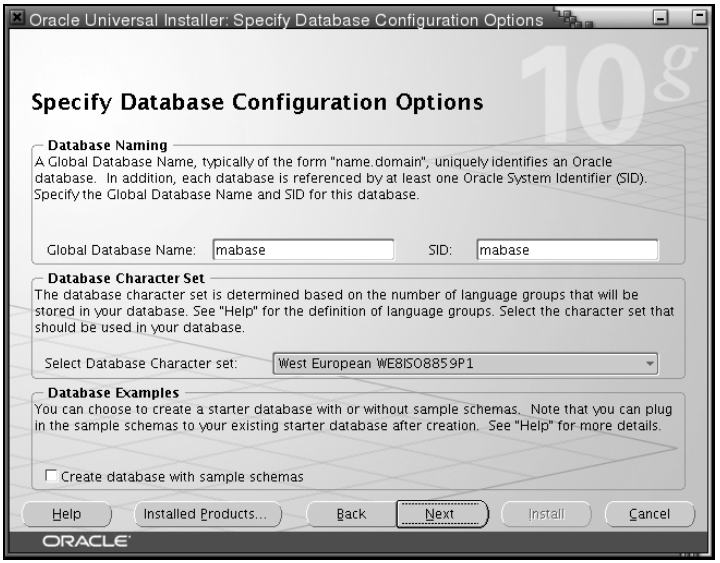


Figure 10.27 : *Paramètres de la base*

Il vous est maintenant demandé de donner un nom à cette base de données un nom interne `SID` et un nom externe `global` (dans notre cas, nous avons simplement choisi le terme "`mabase`"). Cliquez sur **Next**.

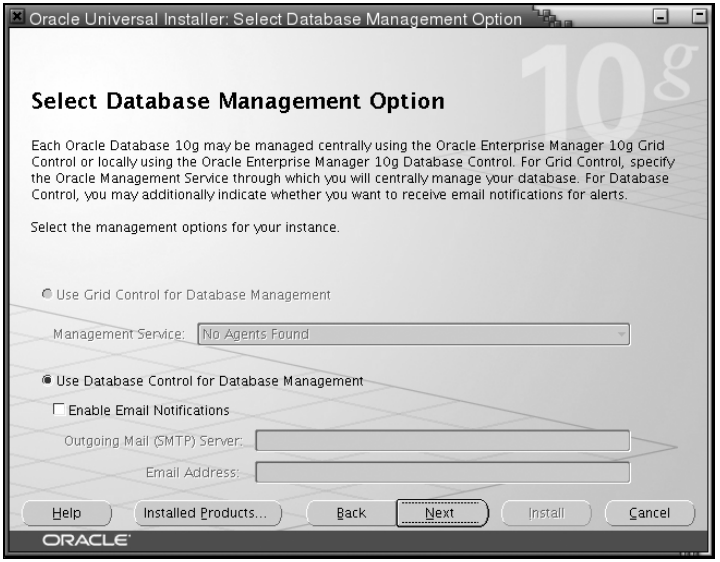


Figure 10.28 : *Options de gestion de la base*

Passons sur les possibilités de gestion de la base. Cliquez sur **Next**.

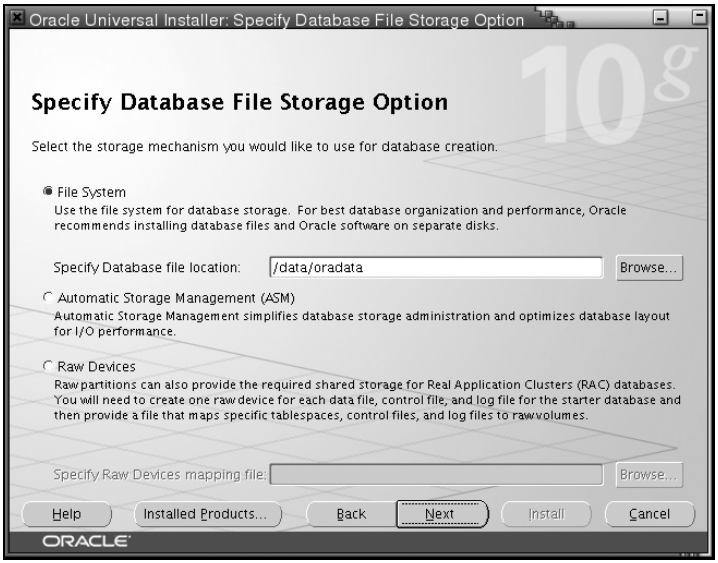


Figure 10.29 : *Emplacement des données*

Il est temps de préciser où doivent être stockées les données de la base de données. Nous avons choisi */data/orainst*. Cliquez sur **Next**.

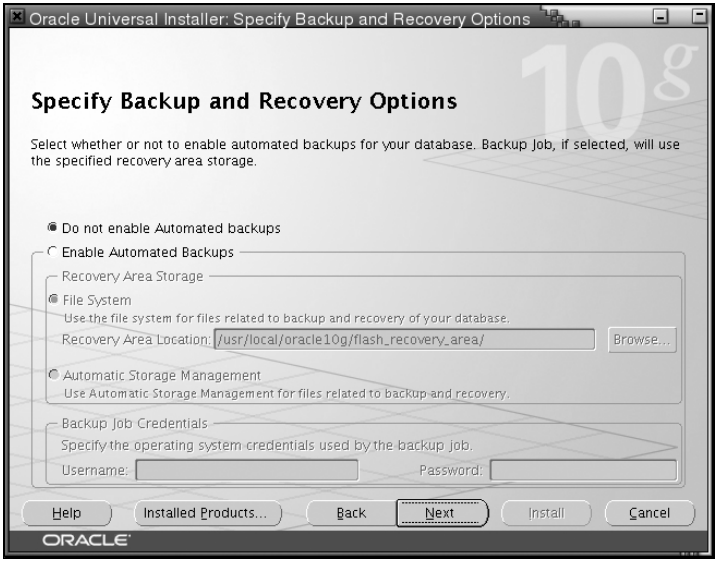


Figure 10.30 : *Possibilités de sauvegarde*

Passons sur les possibilités de sauvegarde (backup). Cliquez sur **Next**.

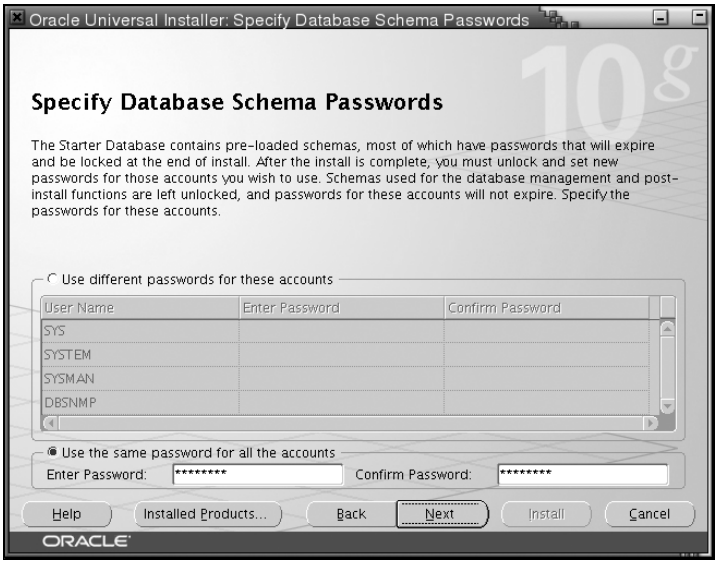


Figure 10.31 : *Spécification des mots de passe*

Par soucis de simplicité, nous avons choisi d'utiliser le même mot de passe pour tous les comptes; à savoir "biblephp" (ce que nous retrouverons dans les fichiers de configuration des scripts). Cliquez sur **Next**. Un premier traitement se déclenche alors.

Figure 10.32 : *Récapitulatif des paramétrages choisis*

Cette fois, l'installation est imminente. Juste le temps de vérifier que toutes les conditions sont remplies. En l'occurrence, la copie d'écran montre un cas d'erreur (espace disque insuffisant) assez facilement corrigé. Cliquez sur **Install**.

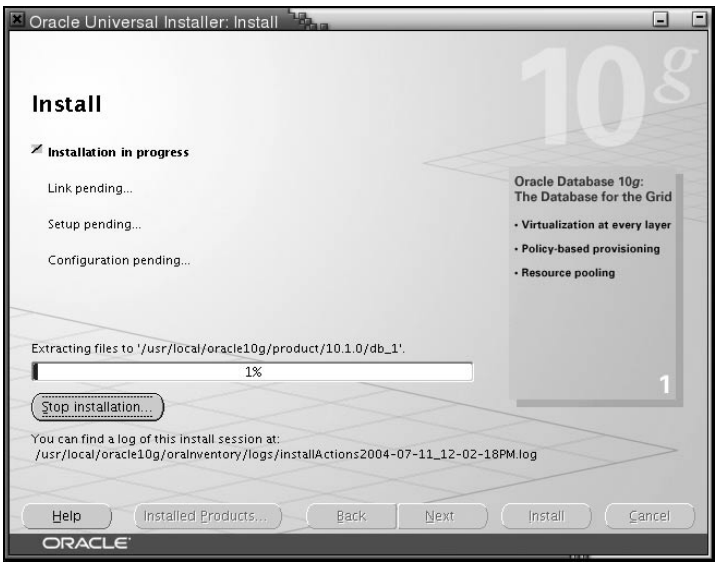


Figure 10.33 : *En cours d'installation*

Cette fois c'est parti.

L'installation risque d'être perturbée par quelques messages d'erreur comme suit mais ne paniquez pas, ils ne devraient pas porter à conséquence. Cliquez simplement sur **Continue**.

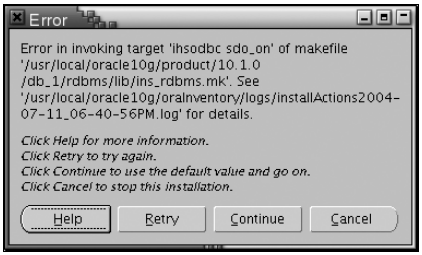


Figure 10.34 :
Message d'erreur lors de l'installation

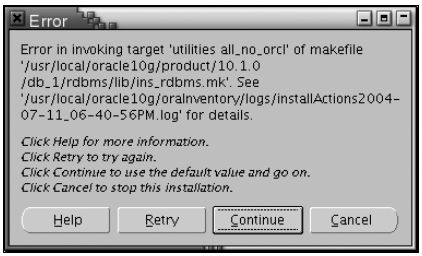


Figure 10.35 :
Ne pas tenir compte des messages d'erreur et continuer

Le logiciel d'installation passe alors à la configuration des différents éléments.

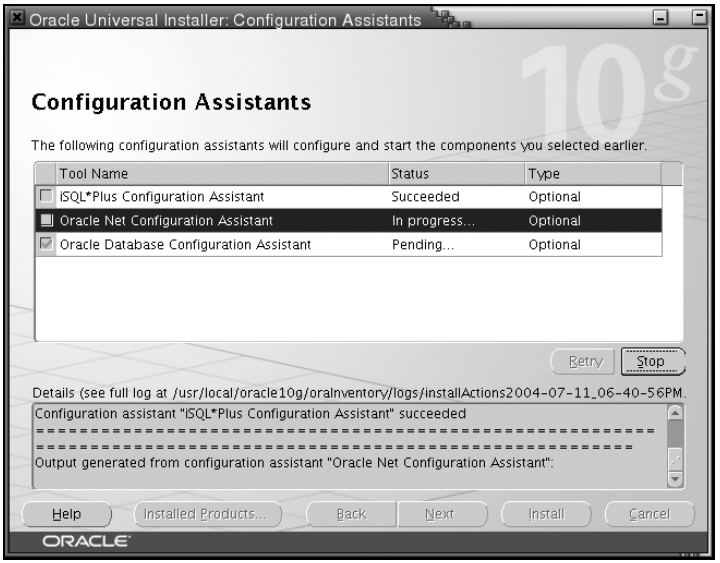


Figure 10.36 : *Les assistants poursuivent l'installation sans votre intervention*

Vous n'avez rien à faire, les opérations s'enchaînent d'elles même.

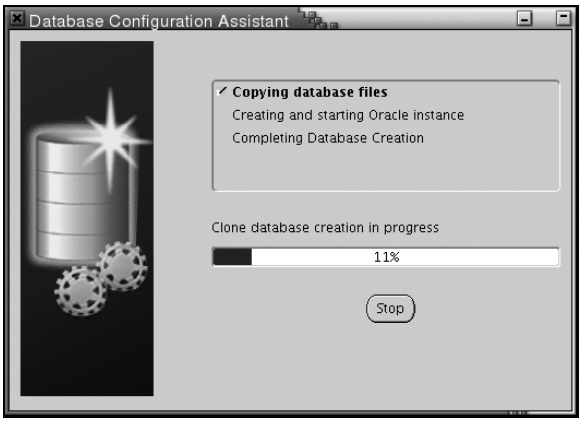


Figure 10.37 : *La base de données se crée enfin*

La base de données se crée.

Complément d'installation

Afin que la base de données (mabase) soit systématiquement démarrée lorsque l'on lance Oracle, vous devez modifier le fichier `/etc/oratab.conf` afin de remplacer le N à la fin de la ligne par un Y.

```
mabase:/usr/local/oracle10g/product/10.1.0/db_1:Y
```

De plus vous êtes invités à modifier le fichier *listener.ora* dans le répertoire *$ORACLE_HOME/network/admin* afin d'ajouter les lignes relatives à `mabase`. Le fichier ressemblera alors à :

```
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = PLSExtProc)
      (ORACLE_HOME = /usr/local/oracle10g/product/10.1.0/db_1)
      (PROGRAM = extproc)
    )
    (SID_DESC =
      (GLOBAL_DBNAME = mabase)
      (ORACLE_HOME = /usr/local/oracle10g/product/10.1.0/db_1)
      (SID_NAME = mabase)
    )
  )

LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST=
        (ADDRESS = (PROTOCOL = IPC)(KEY = EXTPROC))
      )
      (ADDRESS_LIST=
        (ADDRESS = (PROTOCOL = TCP)(HOST=192.168.1.1)(PORT=1521))
      )
    )
  )
```

Ce fichier indique principalement que le *listener* doit être à l'écoute en `TCP` sur la machine `192.168.1.1` (la machine du serveur de bases de données) sur le port 1521 (port par défaut).

Il indique, en outre, qu'il y a sur ce serveur une base de donnée "`mabase`", hébergée sous */usr/local/oracle10g/product/10.1.0/db_1*, et qui sera identifiée comme étant "`mabase`" (par les clients).

Mais il est où ce serveur ?
Afin de vous éviter d'inutiles soucis, si vous indiquez un nom de serveur plutôt qu'une adresse IP assurez qu'Oracle n'a pas de mal à l'identifier. Le plus simple étant alors de déclarer ce serveur dans le fichier /etc/hosts.

Démarrage du serveur de bases de données

Pour démarrer l'interface qui va permettre de communiquer avec le serveur Oracle (et que l'on vient de configurer via *listener.ora*), lancez (sous le compte `oracle` et avec les variables d'environnement définies dans le fichier *.bashrc*):

```
$ lsnrctl start
```

Pour démarrer le serveur Oracle, suivez les instructions suivantes :

```
$ dbstart
```

Test du serveur Oracle (en local)

Vous pouvez maintenant tester la connexion au serveur Oracle.

```
$ sqlplus system/biblephp@mabase
SQL > CREATE TABLE test (id INTEGER);
SQL > INSERT INTO test VALUES (1234);
SQL > SELECT * FROM test;
SQL > DROP TABLE test;
```

Si vous avez pu enchaîner les requêtes précédentes sans générer d'erreur, c'est probablement que le serveur fonctionne correctement.


REMARQUE

Utilisateurs par défaut
Le `system/biblephp` indiqué est un couple `<nom d'utilisateur>/<mot de passe>`. Or lors de l'installation nous avons choisi d'associer le mot de passe `biblephp` à tous les comptes.

Arrêt du serveur de bases de données

Sachez que vous pourrez arrêter Oracle (après avoir mis fin à toutes les connexions, qu'elles aient été ouvertes par sql/plus ou via PHP) en tapant les commandes :

```
$ dbshut
$ lsnrctl stop
```

Configuration de PHP avec support d'Oracle

Sous Linux

Pour accéder à une base de données Oracle depuis PHP, vous n'avez qu'à recompiler PHP avec l'option `--with-oci8=$ORACLE_HOME` (en remplaçant toutefois `$ORACLE_HOME` par sa valeur).


RENVOI

Vous pouvez vous reporter au chapitre "Installation" pour plus de détails sur la façon de compiler PHP.

Il se peut que, durant la compilation de PHP, vous aperceviez un message d'alerte vous invitant à compiler Apache (si tel est votre serveur) avec l'option *pthread* en cas de problème (ce qui n'a pas été notre cas).

Vérification


ATTENTION

Variables d'environnement
Afin de pouvoir accéder à la base de données, le serveur web doit avoir dans ses variables d'environnement celles définies pour le compte `oracle` *(*`ORACLE_HOME`*,* `ORACLE_SID`*, etc.). Pour cela, vous devez intégrer un appel* `source ~oracle/.bashrc` *dans le .bashrc du compte sous lequel tourne le serveur web, ou simplement faire manuellement cet appel juste avant de lancer (dans la même fenêtre) le serveur web.*

Vous pouvez maintenant tester un script ne contenant que le code `<?php phpinfo(); ?>` et vous devez apercevoir les lignes suivantes :

oci8

OCI8 Support	enabled
Revision	$Revision: 1.169.2.3 $
Oracle Version	8.1
Compile-time ORACLE_HOME	/mnt/dba/oracle_home
Libraries Used	

Figure 10.38 : *phpinfo()*

Utilisation

L'utilisation basique de la base de données s'opère selon le schéma suivant :

- Connexion grâce aux fonctions `oci_connect()`, `oci_pconnect()` ou `oci_new_connect()`.
- Soumission de la requête via les fonctions `oci_parse()` et `oci_execute()`.
- Validation ou annulation de la transaction via les fonctions `oci_commit()` ou `oci_rollback()` (si la requête n'a pas été automatiquement validée).
- Déconnexion grâce à la fonction `oci_close()` (dans le cas d'une connexion non persistante).


REMARQUE

A l'époque de PHP 4
La plupart des fonctions décrites ici existaient déjà sous PHP 4 mais portaient un autre nom. Pour chacune d'entre elles nous indiquerons donc leur précédente identité. Il est à noter que pour des raisons de compatibilité, il reste possible d'utiliser l'ancien nom sous PHP 5.

Connexion

La connexion à une base de données Oracle s'opère grâce à la fonction `oci_connect()`.

oci_connect() (ociLogon() PHP4)

Établit une connexion (non persistante) avec le serveur de bases de données.

Syntaxe	`resource oci_connect(string $utilisateur, string $motDePasse [, string $base])`
`$utilisateur`	Nom d'utilisateur.
`$motDePasse`	Mot de passe de l'utilisateur.
`$base`	Nom de la base (`ORACLE_SID`). Si le serveur web et la base de données sont sur la même machine et que le *listener* ne tourne pas, alors vous devez laisser ce champ vide.
`retour`	Identifiant de connexion au serveur de bases de données, ou `FALSE` en cas d'échec.

Si, au sein d'un même script, vous décidez d'ouvrir plusieurs connexions avec `oci_connect()`, vous aurez alors peut-être la surprise de constater que les opérations de `commit` et `callback` s'appliquent à toutes les transactions effectuées, quelle que soit la connexion sélectionnée. Si ce n'est pas le comportement que vous cherchez à obtenir, vous devez alors privilégier la fonction `oci_new_connect()`.

oci_new_connect() (ociNLogon() PHP4)

Établit une nouvelle connexion (non persistante) avec le serveur de bases de données ne partageant pas les opérations de `commit` et `callback` avec les autres connexions ouvertes au sein du script.

Syntaxe	`resource oci_new_connect(string $utilisateur, string $motDePasse [, string $base])`
`$utilisateur`	Nom d'utilisateur.
`$motDePasse`	Mot de passe de l'utilisateur.
`$base`	Nom de la base (`ORACLE_SID`). Si le serveur web et la base de données sont sur la même machine et que le *listener* ne tourne pas, alors vous devez laisser ce champ vide.
`retour`	Identifiant de connexion au serveur de bases de données, ou `FALSE` en cas d'échec.

L'utilisation d'une connexion persistante requiert l'appel à `oci_pconnect()`.

oci_pconnect() (ociPLogon() PHP4)

Établit une connexion persistante avec le serveur de bases de données.

Syntaxe	`resource oci_pconnect(string $utilisateur, string $motDePasse [, string $base])`
`$utilisateur`	Nom d'utilisateur.
`$motDePasse`	Mot de passe de l'utilisateur.
`$base`	Nom de la base (`ORACLE_SID`). Si le serveur web et la base de données sont sur la même machine et que le *listener* ne tourne pas, alors vous devez laisser ce champ vide.
`retour`	Identifiant de connexion au serveur de bases de données, ou `FALSE` en cas d'échec.

Exécution de la requête SQL

L'exécution d'une requête SQL se passe au minimum en deux temps par un appel à `oci_parse()` suivi de `oci_execute()`.

oci_parse() (ociParse() PHP4)

Analyse et prépare une requête SQL.

Syntaxe	`resource oci_parse(resource $idBaseDonnees, string $requete)`
`$idBaseDonnees`	Identifiant de connexion à une base de données tel que retourné par `oci_connect()`, `oci_new_connect()` ou `oci_pconnect()`.
`$requete`	Requête SQL
`retour`	Identifiant de requête SQL, ou `FALSE` en cas d'échec (ex. : identifiant de base de données non valide, mais pas dans le cas d'une requête SQL non valide).

Nous verrons par la suite qu'il est également possible d'utiliser des requêtes préparées (contenant des paramètres qui pourront être modifiés juste avant que celles-ci ne soient exécutées).

oci_execute() (ociExecute() PHP4)

Exécute une requête SQL.

Syntaxe	`boolean oci_execute(resource $idRequete [,int $modeCommit])`
`$idRequete`	Identifiant de requête SQL tel que retourné par `oci_parse()`.
`$modeCommit`	Précise si vous souhaitez que la requête soit immédiatement validée ou non. Vous avez le choix entre les valeurs : `OCI_COMMIT_ON_SUCCESS` (valeur par défaut) : la requête est immédiatement validée (par un `COMMIT`).

	`OCI_DEFAULT` : la requête devra être ultérieurement validée par `oci_commit()` ou annulée par `oci_rollback()`.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

Option OCI_COMMIT_ON_SUCCESS
Ne vous laissez pas surprendre : si vous choisissez l'option (par défaut) OCI_COMMIT_ON_SUCCESS, un `COMMIT` sera nécessairement appliqué (validant ainsi les précédentes opérations effectuées avec l'option OCI_DEFAULT). Ceci est vrai même si la requête effectuée n'est pas une requête de mise à jour de la base (ex. : un `SELECT`).

La requête ainsi exécutée peut être de n'importe quel type. Ce peut être un simple `INSERT` ou `SELECT`, comme une requête de création de procédure stockée ou un appel à une procédure stockée.

Dans le cas d'une requête retournant un résultat (typiquement une requête `SELECT`), il faudra également analyser le résultat. Mais nous verrons cela un peu plus loin.

Commit/rollback

Si vous avez opté pour l'option *OCI_DEFAULT* de la fonction `oci_execute()`, vous pourrez valider ou annuler vos transactions avec les fonctions suivantes :

oci_commit() (ociCommit() PHP4)

Valide les transactions.

Syntaxe	`boolean oci_commit(resource $idBaseDonnees)`
`$idBaseDonnees`	Identifiant de connexion à une base de données tel que retourné par `oci_connect()`, `oci_new_connect()` ou `oci_pconnect()`.
`retour`	`TRUE` en cas de succès, `FALSE` sinon (c'est essentiellement le cas pour un identifiant non valide).

oci_rollback() (ociRollback() PHP4)

Annule les transactions.

Syntaxe	`boolean oci_rollback(resource $idBaseDonnees)`
`$idBaseDonnees`	Identifiant de connexion à une base de données tel que retourné par `oci_connect()`, `oci_new_connect()` ou `oci_pconnect()`.

`retour`	`TRUE` en cas de succès, `FALSE` sinon (c'est essentiellement le cas pour un identifiant non valide).

Déconnexion

Pour vous déconnecter – opération théoriquement facultative mais toutefois vivement conseillée –, vous disposez de la fonction `oci_close()`.

oci_close() (ociLogoff() PHP4)

Met fin à la connexion à la base de données et libère les ressources associées.

Syntaxe	`void oci_close(resource $idBaseDonnees)`
`$idBaseDonnees`	Identifiant de connexion à une base de données tel que retourné par `oci_connect()`, `oci_new_connect()` ou `oci_pconnect()`.

Premier exemple

Nous voici donc à même de construire notre premier script utilisant une base de données.

Vérifiez les paramètres
Prenez garde ! Avant d'exécuter ce script, assurez-vous que les paramètres prédéfinis ne risquent pas de conduire à la suppression des données d'une de vos bases qui, par coïncidence, existerait déjà.

Comme cela est fortement conseillé, nous avons, ici, isolé les paramètres de connexion dans un fichier aisément repérable.

Listing 10.58 : parametres_bd_inc.php

```
<?php

    // Paramètres de connexion à la base de données
    // N'hésitez pas à les modifier selon vos besoins

    $utilisateur = "system";
    $motDePasse  = "manager";

    $base        = "mabase";
?>
```

Notre script principal sera donc :

Listing 10.59 : insert01.php

```
<?php

    // Paramètres du script
    require_once("parametres_bd_inc.php");
    $table = "tableexemple";

    // inclusion du script contenant les fonctions
    require_once("insert01_bd_inc.php");

    // Connexion à la base de données
    $idConnexion = oci_pconnect($utilisateur, $motDePasse, $base);
    if (!$idConnexion) {
        die("<b>Impossible de se connecter à la base de données".
            " [$base] sous le compte [$utilisateur]".
            " avec le mot de passe [$motDePasse].</b>");
    }


    // Appel de la fonction principale
    if (EX_initialiseBD($idConnexion, $table)) {
        echo "Voilà, une nouvelle table avec quelques données,".
             "vous pouvez le vérfier avec votre client de base ".
             " de données ou via les scripts suivants.";
    } else {
         echo "La création ou l'alimentation de la table à échouée.";
    }

    // Pas de déconnexion dans le cas d'une connexion persistante
    // oci_close($idConnexion);
?>
```

et nécessitera :

Listing 10.60 : insert01_bd_inc.php

```
<?php

    /** Fonction chargé de créer et d'alimenter une table
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la table
     **/
   function EX_initialiseBD($idConnexion, $table)
   {
       // Création de la table
```

```
        $requete = "DROP TABLE $table";
        $idRequete = oci_parse($idConnexion, $requete);
        @oci_execute($idRequete);

        $requete = "CREATE TABLE $table (filmId INTEGER, film VARCHAR(64))";
        $idRequete = oci_parse($idConnexion, $requete);
        if (!oci_execute($idRequete)) return FALSE;

        // Création d'une séquence pour les identifiants de film
        $requete = "DROP SEQUENCE seq_$table";
        $idRequete = oci_parse($idConnexion, $requete);
        @oci_execute($idRequete);

        $requete = "CREATE SEQUENCE seq_$table";
        $idRequete = oci_parse($idConnexion, $requete);
        if (!oci_execute($idRequete)) return FALSE;

        // Création d'un trigger pour remplir automatiquement
        // le champ identifiant de film à chaque ajout de film
        $requete = "DROP TRIGGER compteur_$table";
        $idRequete = oci_parse($idConnexion, $requete);
        @oci_execute($idRequete);

        $requete = "CREATE TRIGGER compteur_$table BEFORE INSERT ON $table ".
                   "FOR EACH ROW ".
                   "BEGIN ".
                   "SELECT seq_$table.NEXTVAL INTO :NEW.filmId FROM DUAL; ".
                   "END;";
        $idRequete = oci_parse($idConnexion, $requete);
        if (!oci_execute($idRequete)) return FALSE;

        // Ajoute quelques données
        $idRequete = oci_parse($idConnexion,
                     "INSERT INTO $table (film) VALUES ('Forrest Gump')");
        oci_execute($idRequete, OCI_DEFAULT);

        $idRequete = oci_parse($idConnexion,
                     "INSERT INTO $table (film) VALUES ('Matrix')");
        oci_execute($idRequete, OCI_DEFAULT);

        $idRequete = oci_parse($idConnexion,
                   "INSERT INTO $table (film) VALUES ('La cité de la peur')");
        oci_execute($idRequete, OCI_DEFAULT);

        // Validation des transactions
        // Uniquement parce que l'option OCI_DEFAULT a été choisie.
        oci_commit($idConnexion);

        return TRUE;
}
?>
```

Lecture des enregistrements

Dans le cas d'une requête retournant une liste de données (typiquement un `SELECT`), il convient de récupérer l'identifiant de requête et de l'utiliser pour lire, en boucle, ligne après ligne, chaque enregistrement. Il existe diverses fonctions, chacune permettant de récupérer les champs des enregistrements sous différentes formes.

Les informations peuvent ainsi être retournées :

- Champ par champ ;
- Enregistrement par enregistrement (sous la forme d'un tableau indexé, d'un tableau associatif ou encore d'un tableau à la fois indexé et associatif) ;
- Par bloc d'enregistrements.

Lecture champ par champ

Pour une lecture champ par champ, vous pouvez passer d'un enregistrement au suivant par l'appel à `oci_fetch()`, et accéder à l'un quelconque des champs par appel à `oci_result()`.

oci_fetch() (ociFetch() PHP4)

Passe à l'enregistrement suivant.

Syntaxe	`boolean oci_fetch(resource $idRequete)`
`$idRequete`	Identifiant de requête tel que retourné par `oci_execute()`.
`retour`	`TRUE` en cas de succès, `FALSE` sinon (plus d'enregistrement à lire).

oci_result() (ociResult() PHP4)

Retourne la valeur d'un champ d'un enregistrement.

Syntaxe	`mixed oci_result(resource $idRequete, mixed $champ)`
`$idRequete`	Identifiant de requête tel que retourné par `oci_execute()`.
`$champ`	Au choix, soit le nom du champ (en majuscules), soit l'index du champ dans la requête (le premier champ ayant l'index 0).
`retour`	La valeur du champ.

Listing 10.61 : select_cc_bd_inc.php

```
<?php

    /**
     * Fonction listant le contenu d'une table
```

```
     * contenant 2 champs (filmId et film)
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la table
     **/

    function EX_listeContenu($idConnexion, $table)
    {
        // Requête
        $requete = "SELECT * FROM $table";
        $idRequete = oci_parse($idConnexion, $requete);
        if (!oci_execute($idRequete)) return FALSE;

        // Boucle de lecture (et d'affichage) des enregistrements
        while (oci_fetch($idRequete, $enreg)) {
            echo "FilmId=".oci_result($idRequete, "FILMID")." ";
            echo "Film=".oci_result($idRequete, "FILM")."<br />";
        }
        return TRUE;
    }
?>
```


Lecture enregistrement par enregistrement

La forme la plus simple est celle retournée par la fonction `oci_fetch_row()`.

oci_fetch_row()

Retourne un enregistrement, retourné par une requête SQL, sous la forme d'un tableau indexé.

Syntaxe	`array oci_fetch_row(resource $idRequete)`
`$idRequete`	Identifiant de requête tel que retourné par `oci_execute()`.
`retour`	Tableau indexé contenant autant d'éléments que de champs retournés par la requête.

Listing 10.62 : select_eei_bd_inc.php

```
<?php

    /**
     * Fonction listant le contenu d'une table
     * contenant 2 champs (filmId et film)
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la table
     **/

    function EX_listeContenu($idConnexion, $table)
```

```
    {
        // Requête
        $requete = "SELECT * FROM $table";
        $idRequete = oci_parse($idConnexion, $requete);
        if (!oci_execute($idRequete)) return FALSE;

        // Boucle de lecture (et d'affichage) des enregistrements
        while ($enreg = oci_fetch_row($idRequete)) {
            echo "FilmId=".$enreg[0]." Film=".$enreg[1]."<br />";
        }
        return TRUE;
    }
?>
```

Mais il est également possible de retourner un enregistrement sous la forme d'un tableau indexé et associatif avec, pour clés, les noms des champs (en majuscules).

oci_fetch_array()

Retourne un enregistrement, retourné par une requête SQL, sous la forme d'un tableau indexé et/ou associatif.

Syntaxe	`array oci_fetch_array(resource $idRequete [, int $mode])`
`$idRequete`	Identifiant de requête tel que retourné par `oci_execute()`.
`$mode`	Combinaison par `OU` logique des options suivantes : *`OCI_ASSOC : l'enregistrement est retourné sous la forme d'un tableau associatif où les clés sont les noms des champs (en majuscules) et les valeurs, celles des champs. Il n'y a pas de clé créée pour les champs ayant une valeur`* `NULL` *`(sauf si l'option est combinée avec`* `OCI_RETURN_NULLS`*`).`* `OCI_NUM` (valeur par défaut) : l'enregistrement est retourné sous la forme d'un tableau indexé (l'index 0 correspondant au premier champ de l'enregistrement). Il n'y a pas d'index créé pour les champs ayant une valeur `NULL` (sauf si l'option est combinée avec `OCI_RETURN_NULLS`). `OCI_BOTH` : combinaison de `OCI_ASSOC` et `OCI_NUM`. `OCI_RETURN_NULLS` : complète les tableaux avec des index ou clés, y compris pour les valeurs `NULL`. `OCI_RETURN_LOBS` : retourne les valeurs des champs de type `CLOB`, `BLOB` et non leurs descripteurs (voir plus loin).
`retour`	Tableau associatif ou indexé selon le mode choisi.

C'est ce mode de lecture de données que nous privilégierons; il est en effet fort pratique pour les champs dont on connaît le nom (ce qui est le plus souvent le cas), et permet également de faire référence à des champs sans nom, comme par exemple un champ `SUM(nomchamp)`. Notez toutefois qu'il est possible d'affecter un nom (ex: *somme*) à ce genre de champ en indiquant `SUM(nomchamp) AS somme`.

L'exemple précédent gagnera ainsi en lisibilité et deviendra:

Listing 10.63 : select_eea_bd_inc.php

```
<?php

    /**
     * Fonction listant le contenu d'une table
     * contenant 2 champs (filmId et film)
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la table
     **/

    function EX_listeContenu($idConnexion, $table)
    {
        // Requête
        $requete = "SELECT * FROM $table";
        $idRequete = oci_parse($idConnexion, $requete);
        if (!oci_execute($idRequete)) return FALSE;

        // Boucle de lecture (et d'affichage) des enregistrements
        while ($enreg = oci_fetch_array($idRequete, OCI_BOTH)) {
            echo "FilmId=".$enreg["FILMID"]." Film=".$enreg["FILM"]."<br />";
        }
        return TRUE;
    }

?>
```

`oci_fetch_array` est l'équivalent de la fonction PHP4 `ociFetchInto()`, (toujours disponible mais amenée à disparaître) mais cette dernière à une syntaxe légèrement différente.

ociFetchInto() (PHP4)

Retourne un enregistrement, retourné par une requête SQL, sous la forme d'un tableau indexé, associatif, ou encore indexé et associatif.

Syntaxe	`boolean ociFetchInto(resource $idRequete, array &$enregistrement [, int $mode])`
`$idRequete`	Identifiant de requête tel que retourné par `ociExecute()`.
`$enregistrement`	Référence sur un tableau dans lequel seront copiées les données de l'enregistrement.
`$mode`	Combinaison par `OU` logique des options suivantes :
	`OCI_ASSOC : l'enregistrement est retourné sous la forme d'un tableau associatif où les clés sont les noms des champs (en majuscules) et les valeurs, celles des champs. Il n'y a pas de clé`

	créée pour les champs ayant une valeur NULL (sauf si l'option est combinée avec OCI_RETURN_NULLS). OCI_NUM (valeur par défaut) : l'enregistrement est retourné sous la forme d'un tableau indexé (l'index 0 correspondant au premier champ de l'enregistrement). Il n'y a pas d'index créé pour les champs ayant une valeur NULL (sauf si l'option est combinée avec OCI_RETURN_NULLS). OCI_BOTH : combinaison de OCI_ASSOC et OCI_NUM. OCI_RETURN_NULLS : complète les tableaux avec des index ou clés, y compris pour les valeurs NULL. OCI_RETURN_LOBS : retourne les valeurs des champs de type CLOB, BLOB et non leurs descripteurs (voir plus loin).
retour	TRUE en cas de succès, FALSE sinon (plus d'enregistrement à lire).

Lecture des enregistrements en un bloc

Il est également possible de retourner l'ensemble des enregistrements par un appel unique à la fonction `oci_fetch_all()` (ce qui n'est pas nécessairement la façon la plus élégante de procéder).

oci_fetch_all() (ociFetchStatement() PHP4)

Retourne l'ensemble des enregistrements sous la forme d'un tableau.

Syntaxe	`int oci_fetch_all(resource $idRequete, array &$tableauEnregistrements)`
`$idRequete`	Identifiant de requête tel que retourné par `oci_execute()`.
`$tableauEnregistrements`	Référence sur une variable dans laquelle sera copié un tableau associatif ayant pour clés les champs (en majuscules) des enregistrements, et pour valeur un tableau indexé contenant les valeurs pour chaque enregistrement.
`retour`	Nombre d'enregistrements retournés.

Association des champs retournés par une requête à des variables PHP

Il est également possible de lire le résultat d'une requête SQL en associant directement un champ de l'enregistrement à une variable PHP. Pour cela, vous pouvez utiliser la fonction `oci_define_by_name()`.

oci_define_by_name() (ociDefineByName() PHP4)

Associe à une variable PHP un champ retourné par une requête (de type SELECT).

Syntaxe	boolean oci_define_by_name(resource $idRequete, string $champ, mixed &$variable [, int $typeChamp])
$idRequete	Identifiant de requête tel que retourné par oci_execute().
$champ	Nom du champ (en majuscules) de l'enregistrement.
$variable	Référence sur la variable PHP devant être liée à la variable SQL.
$typeChamp	Type compatible avec le champ SQL. Ce paramètre peut prendre une valeur parmi les suivantes : OCI_B_BFILE. OCI_B_CFILEE. OCI_CLOB s'il s'agit d'un champ de type CLOB (objet large de type texte). OCI_BLOB s'il s'agit d'un champ de type BLOB (objet large de type binaire). OCI_ROWID s'il s'agit d'un identifiant d'enregistrement. OCI_B_CURSOR s'il s'agit d'un curseur (pointeur de résultat). OCI_B_BIN. OCI_B_SQLT_NTY s'il s'agit d'une collection. OCI_SYSDATE. Par défaut, le champ est retourné sous le type string.
retour	Le contenu ou pointeur vers le champ.

Si vous avez pris soin d'appeler cette fonction avant de faire un appel à oci_execute(), alors, à chaque appel à oci_fetch(), les contenus des variables ainsi associées aux champs de l'enregistrement seront mis à jour avec les valeurs de l'enregistrement suivant.

Listing 10.64 : select_bind_bd_inc.php

```
<?php

    /**
     * Fonction listant le contenu d'une table
     * contenant 2 champs (filmId et film)
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la table
     **/

    function EX_listeContenu($idConnexion, $table)
    {
        // Requête
        $requete = "SELECT * FROM $table";
        $idRequete = oci_parse($idConnexion, $requete);
        oci_define_by_name($idRequete, "FILMID", $filmId);
        oci_define_by_name($idRequete, "FILM", $film);
        if (!oci_execute($idRequete)) return FALSE;
```

```
        // Boucle de lecture (et d'affichage) des enregistrements
        while (oci_fetch($idRequete)) {
            echo "FilmId=".$filmId." Film=".$film."<br />";
        }
        return TRUE;
    }

?>
```

Si le champ associé est de type `BLOB` ou `CLOB`, vous devrez faire appel, au préalable, à `oci_new_descriptor()`, comme cela est décrit dans le chapitre sur les objets de grande taille.

Nombre d'enregistrements

Nombre d'enregistrements retournés

Si l'on inclus l'utilisation de `ociFetchStatement()` qui retourne un tableau dont il est aisé de connaître la taille, il existe trois façons de déterminer le nombre d'enregistrements retournés par une requête.

Si vous souhaitez uniquement connaître le nombre d'enregistrements, mais ne souhaitez pas immédiatement lire ces enregistrements, alors il vous suffit de faire une requête `COUNT()` comme suit :

Listing 10.65 : count_bd_inc.php

```
<?php

    /**
     * Fonction affichant le nombre d'enregistrements
     * dans une table
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la tabme
     **/

    function EX_compte($idConnexion, $table)
    {
        // Requete
        $requete = "SELECT COUNT(*) FROM $table";
        $idRequete = oci_parse($idConnexion, $requete);
        if (!oci_execute($idRequete)) return FALSE;

        // Lecture du résultat
        if ($enreg = oci_fetch_row($idRequete)) {
            echo "Il y a ".$enreg[0]." enregistrements dans".
                 " la table $table<br />";
            return TRUE;
        } else {
            echo "Etes-vous sûr que la table $table existe ?<br />";
```

```
            return FALSE;
        }
    }
?>
```

Si vous souhaitez connaître le nombre d'enregistrements, mais que vous souhaitez lire les enregistrements sans pour autant avoir besoin au préalable d'en connaître le nombre, il suffit de les compter au fur et à mesure de leur lecture.

Listing 10.66 : count_select_bd_inc.php

```
<?php

    /**
     * Fonction affichant le nombre d'enregistrements
     * dans une table
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la tabme
     **/

    function EX_compte($idConnexion, $table)
    {
        // Requete
        $requete = "SELECT * FROM $table";
        $idRequete = oci_parse($idConnexion, $requete);
        if (!oci_execute($idRequete)) return FALSE;

        // Boucle de lecture (et de décompte) des enregistrements
        $nb = 0;
        while ($enreg = oci_fetch_row($idRequete)) {
            // vous pouvez manipuler $enreg a votre guise
            // mais n'oubliez pas de le compter
            $nb++;
        }

        echo "Il y a $nb enregistrements dans la table $table<br />";
        return TRUE;
    }
?>
```

Sachant qu'une requête `SELECT *` est plus longue qu'une requête `SELECT COUNT(*)`, privilégiez la première méthode si vous n'avez que faire du contenu des enregistrements.

Nombre d'enregistrements modifiés

Il est possible de déterminer le nombre d'enregistrements modifiés (par une requête de type `INSERT` ou `UPDATE`, mais à l'exclusion des enregistrements retournés par `SELECT`) avec la fonction `oci_num_rows()`.

oci_num_rows() (ociRowCount() PHP4)

Retourne le nombre d'enregistrements modifiés par la requête.

Syntaxe	`int oci_num_rows(resource $idRequete)`
`$idRequete`	Identifiant de requête tel que retourné par `oci_execute()`.
`retour`	Nombre d'enregistrements modifiés.

Dans notre exemple, nous modifierons les enregistrements sans véritablement les modifier.

Listing 10.67 : count_update_bd_inc.php

```
<?php

    /**
     * Fonction affichant le nombre d'enregistrements
     * dans une table
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la table
     **/

    function EX_compte($idConnexion, $table)
    {
        // Requete
        $requete = "UPDATE $table SET filmId=filmId+0";
        $idRequete = oci_parse($idConnexion, $requete);
        if (!oci_execute($idRequete)) return FALSE;

        echo "Il y a eu ".oci_num_rows($idRequete).
             " enregistrements modifiés<br />";
        return TRUE;
    }
?>
```

Mise à profit des requêtes préparées

Oracle permet l'analyse, la compilation et le stockage des requêtes avant utilisation. Ceci permet d'exécuter une série de requêtes similaires, sans avoir à renouveler à chaque fois les opérations d'analyse et de compilation.

Pour cela, il suffit de préparer une requête dans laquelle les éléments variables sont remplacés par des noms précédés de : (ex. : `INSERT INTO matable (film) VALUES (:film)`), et d'associer cette variable SQL à une variable PHP.

Cette association variable SQL/variable PHP se réalise par la fonction `oci_bind_by_name()`.

oci_bind_by_name() (ociBindByName() PHP4)

Associe une variable de requête SQL à une variable PHP.

Syntaxe	`boolean oci_bind_by_name(resource $idRequete, string $variableSQL, mixed &$variablePHP [, int $longueurChamp , [int $typeChamp]])`
`$idRequete`	Identifiant de requête tel que retourné par `oci_execute()`.
`$variableSQL`	Nom de la variable dans la requête SQL.
`$variablePHP`	Référence sur la variable PHP devant être liée à la variable SQL.
`$longueurChamp`	Longueur du champ SQL.
`$typeChamp`	Type du champ SQL. Ce paramètre peut prendre une valeur parmi les suivantes : `OCI_B_BFILE`. `OCI_B_CFILEE`. `OCI_CLOB` s'il s'agit d'un champ de type `CLOB` (objet large de type texte). `OCI_BLOB` s'il s'agit d'un champ de type `BLOB` (objet large de type binaire). `OCI_ROWID` s'il s'agit d'un identifiant d'enregistrement. `OCI_B_CURSOR` s'il s'agit d'un curseur (pointeur de résultat). `OCI_B_BIN`. `OCI_B_SQLT_NTY` s'il s'agit d'une collection. `OCI_SYSDATE`.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

En utilisant les requêtes préparées, le script d'initialisation de la table d'exemple devient :

Listing 10.68 : insert02_bd_inc.php

```
<?php

    /** Fonction chargé de créer et d'alimenter une table
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la table
     **/
   function EX_initialiseBD($idConnexion, $table)
   {
       // Création de la table
       $requete = "DROP TABLE $table";
       $idRequete = oci_parse($idConnexion, $requete);
       @oci_execute($idRequete);

       $requete = "CREATE TABLE $table (filmId INTEGER, film VARCHAR(64))";
       $idRequete = oci_parse($idConnexion, $requete);
       if (!oci_execute($idRequete)) return FALSE;

       // Création d'une séquence pour les identifiants de film
```

```
        $requete = "DROP SEQUENCE seq_$table";
        $idRequete = oci_parse($idConnexion, $requete);
        @oci_execute($idRequete);

        $requete = "CREATE SEQUENCE seq_$table";
        $idRequete = oci_parse($idConnexion, $requete);
        if (!oci_execute($idRequete)) return FALSE;

        // Création d'un trigger pour remplir automatiquement
        // le champ identifiant de film à chaque ajout de film
        $requete = "DROP TRIGGER compteur_$table";
        $idRequete = oci_parse($idConnexion, $requete);
        @oci_execute($idRequete);

        $requete = "CREATE TRIGGER compteur_$table BEFORE INSERT ON $table ".
                   "FOR EACH ROW ".
                   "BEGIN ".
                   "SELECT seq_$table.NEXTVAL INTO :NEW.filmId FROM DUAL; ".
                   "END;";
        $idRequete = oci_parse($idConnexion, $requete);
        if (!oci_execute($idRequete)) return FALSE;

        // Prépare une requête
        $idRequete = oci_parse($idConnexion,
                     "INSERT INTO $table (film) VALUES (:film)");

        // Associe une variable SQL à une variable PHP
        oci_bind_by_name($idRequete, ":film", $film, 64);

        // 3 insertion en mode "auto commit"
        $film = 'Forrest Gump';
        if (!oci_execute($idRequete)) return FALSE;

        $film = 'Matrix';
        if (!oci_execute($idRequete)) return FALSE;

        $film = 'La cité de la peur';
        if (!oci_execute($idRequete)) return FALSE;

        return TRUE;
    }
?>
```

Utilisation des objets de grande taille (BLOB, CLOB)

PHP vous permet, bien entendu, de manipuler les champs de type `BLOB` et `CLOB` mis à votre disposition par la base de données Oracle.

Lorsque vous voudrez créer un objet de type `LOB`, vous devrez faire appel à la fonction `oci_new_descriptor()` afin de créer un objet PHP qui vous servira à en manipuler le contenu.

oci_new_descriptor() (ociNewDescriptor() PHP4)

Crée un objet PHP permettant la manipulation des objets de grande taille.

Syntaxe	`object oci_new_descriptor(resource $idConnexion [, int $typeObjet])`
`$idBaseDonnees`	Identifiant de connexion à une base de données tel que retourné par `oci_connect()`, `oci_new_connect()` ou `oci_pconnect()`.
`$typeObjet`	Au choix : `OCI_D_FILE`. `OCI_D_LOB` s'il s'agit d'un objet *LOB*. `OCI_ROWID` s'il s'agit d'un identifiant d'enregistrement.
`retour`	L'objet demandé.

Ces objets de grande taille sont représentés sous PHP par des objets (PHP) possédant un attribut "`descriptor`" contenant un pointeur (`resource`).

Ces objets *OCILOB* proposent différentes méthodes dont les essentielles :

- `free()` pour libérer les ressources allouées par l'objet.
- `load()` pour récupérer le contenu du champ (`BLOB`, `CLOB`) ;
- `save()` pour insérer un objet dans la base ;
- `saveFile()` pour sauvegarder le contenu du champ dans un fichier ;
- `writeToFile()` pour sauvegarder le contenu du champ dans un fichier ;

mais aussi `append()`, `close()`, `eof()`, `erase()`, `flush()`, `read()`, `rewind()`, `seek()`, `setBuffering()`, `size()`, `tell()`, `truncate()`, `write()`, `writeTemporary()`

PHP 5.0.0 pas encore totalement au point ?
Il semblerait qu'il ait été prévu que les méthodes `saveFile()`; `writeToFile()`, `writeTemporary()` (qui existaient déjà sous PHP4) soient renommées en `import()`, `export()`, `write_temporary()` ce n'est toutefois pas le cas avec PHP 5.0.0.

OCILOB->save()

Insère un objet dans un champ de type `LOB`.

Syntaxe	`boolean save(string $donnees)`
`$donnees`	Données constituant l'objet.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

OCILOB->load()

Retourne le contenu d'un champ de type LOB.

Syntaxe	`string load(void)`
`retour`	Contenu du champ, ou `FALSE` en cas d'échec.

OCILOB->saveFile()

Sauve le contenu d'un champ de type LOB dans un fichier.

Syntaxe	`boolean saveFile(string $nomFichier)`
`$nomFichier`	Nom du fichier dans lequel copier le contenu de l'objet.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

OCILOB->writeToFile()

Sauve une portion du contenu d'un champ de type BLOB dans un fichier.

Syntaxe	`boolean writeToFile(string $nomFichier [, int $debut [, int $longueur]])`
`$nomFichier`	Nom du fichier dans lequel copier le contenu de l'objet.
`$debut`	Octet à partir duquel commencer la copie de l'objet.
`$longueur`	Nombre d'octets à écrire dans le fichier.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

OCILOB->free()

Libère les ressources allouées par l'objet.

Syntaxe	`boolean free(void)`
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

Voici un exemple d'utilisation : dans un premier temps, nous ajoutons un objet *BLOB* dans la base, puis nous le récupérons de deux façons différentes pour le sauvegarder dans un fichier ou l'afficher.

Listing 10.69 : blob_bd_inc.php

```
<?php

// Fonction chargé de donner un exemple d'utilisation
// des BLOB
// Compatibilite: PHP 5

function EX_blob($idConnexion, $table)
{
    // Création d'une table de test
    $idRequete = oci_parse($idConnexion, "DROP TABLE $table");
    @oci_execute($idRequete);
    $idRequete = oci_parse($idConnexion, "CREATE TABLE $table (id INTEGER,".
                                                              "monblob BLOB)");
    oci_execute($idRequete);

    //-----------------------------------
    // Insertion d'un objet de type blob
    //-----------------------------------
    $requete = "INSERT INTO $table (id, monblob) VALUES (1, EMPTY_BLOB())".
               " RETURNING monblob INTO :blob";
    $idRequete = oci_parse($idConnexion, $requete);

    // Création d'un descripteur de BLOB
    $blob = oci_new_descriptor($idConnexion, OCI_D_LOB);

    oci_bind_by_name($idRequete, ":blob", $blob, -1, OCI_B_BLOB);
    if (!oci_execute($idRequete, OCI_DEFAULT)) die("echec.");

    // Insertion des données
    // cela nécessite une transaction (d'où le OCI_DEFAULT)
    $blob->save("On supposera qu'il y a là une grande quantité de données".
                " de type binaire (issue par exemple de la lecture d'un ".
                " fichier image)");
    oci_commit($idConnexion);
    echo "Données enregistrées<br />";

    $blob->free();

    //-------------------------------------------------------------
    // Lecture d'un objet de type BLOB par un simple oci_fetch_assoc
    //-------------------------------------------------------------
    $requete = "SELECT monblob FROM $table WHERE id=1";
    $idRequete = oci_parse($idConnexion, $requete);

    if (!oci_execute($idRequete)) return FALSE;

    echo "<b>Lecture avec oci_fetch_assoc</b><br />";
    if (! ($enreg = oci_fetch_assoc($idRequete))) return FALSE;

    $blob = $enreg["MONBLOB"];
```

```
    // Une fois le descripteur récupéré
    // Il est possible de...

    // Récupérer le contenu de l'objet (pour l'afficher)
    echo "<b>Affichage avec la méthode load()</b><br />";
    echo  $blob->load()."<br />";

    // Sauvegarder l'objet dans un fichier
    // $sts = $blob->export("blob.txt");
    $sts = $blob->writeToFile("blob.txt");

    // Sauvegarder une partie de l'objet dans un fichier
    // $sts = $blob->export("blob2.txt", 3, 5);
    $sts = $blob->writeToFile("blob2.txt", 3, 5);

    $blob->free();

    //-------------------------------------------------------------
    // Lecture d'un objet de type BLOB par ociDefineByName
    //-------------------------------------------------------------
    $requete = "SELECT monblob FROM $table WHERE id=1";
    $idRequete = oci_parse($idConnexion, $requete);

    // Création d'un descripteur de BLOB
    $blob = oci_new_descriptor($idConnexion, OCI_D_LOB);

    oci_define_by_name($idRequete, "MONBLOB", $blob, OCI_B_BLOB);

    if (!oci_execute($idRequete)) return FALSE;
    if (!oci_fetch($idRequete)) return FALSE;

    // Une fois le descripteur récupéré
    // Il est possible de...

    // Récupérer le contenu de l'objet (pour l'afficher)
    echo "<b>Lecture avec oci_define_by_name</b><br />";
    echo  $blob->load();

    $blob->free();

    return TRUE;
}
?>
```

L'ajout de l'objet se fait selon les étapes suivantes :

1. Préparation d'une requête d'insertion ajoutant un objet *BLOB* vide et retournant l'objet ;
2. Création d'un objet (PHP) ;

3. Association de cet objet PHP à l'objet SQL ;
4. Exécution proprement dite. Les appels aux méthodes d'écriture de l'objet *BLOB* devant obligatoirement se faire au sein d'une transaction, il est important, ici, de sélectionner le mode `OCI_DEFAULT`.
5. Appel de la méthode `save()` afin de préciser le contenu de l'objet *BLOB* ;
6. Et enfin, validation de la transaction avec `oci_commit()` ;
7. Libération des ressources.

La lecture de l'objet *BLOB* peut se faire de différentes façons :

- Si vous faites appel à `oci_fetch_assoc()`, `oci_fetch_row()` ou encode `oci_fetch_array()` (comme avec un champ de type standard), vous récupérez alors un objet PHP, sur lequel vous pouvez appeler directement la méthode voulue.
- Si vous faites appel à `oci_bind_by_name()`, vous devrez, au préalable, allouer les ressources de l'objet PHP associé avec `oci_new_descriptor()`.

Gestion des erreurs

Jusque-là, en cas d'erreur, nous nous sommes contentés d'afficher un message générique. Il est cependant possible de déterminer plus précisément l'origine de l'erreur.

Cela est possible notamment en récupérant un code d'erreur via la fonction `oci_error()`.

oci_error() (ociError() PHP4)

Retourne le code d'erreur du dernier appel à la connexion ou à la requête.

Syntaxe	`array oci_error([resource $idBaseDonneesOuRequete])`
`$idBaseDonnees OuRequete`	Identifiant de connexion à une base de données tel que retourné par `oci_connect()`, `oci_new_connect()` ou `oci_pconnect()`, ou identifiant de requête tel que retourné par `oci_execute()`. Si ce paramètre n'est pas précisé, c'est la dernière erreur rencontrée, et non liée à un identifiant de connexion ou de requête (ex. : problème de connexion), qui est retournée.
`retour`	Tableau associatif contenant la clé "`code`", à laquelle est associé un code d'erreur (un entier) et la clé "`message`" à laquelle est associé un message d'erreur (une chaîne de caractères en anglais). `FALSE` en cas d'absence d'erreur.

Tableau 10.16 : Exemples de codes et messages d'erreur

Code	Message	Description
900	*ORA-00900: invalid SQL statement*	Requête SQL non valide.
1017	*ORA-01017: invalid username/password; logon denied*	L'identifiant de connexion à la base n'est pas valide.
12154	*ORA-12154: TNS:could not resolve service name*	La connexion à la base (via le *listener*) n'a pas pu s'effectuer.

Vous pourrez, par exemple, vous servir des codes d'erreur pour afficher vos propres messages d'erreur. Dans l'exemple suivant, nous montrons comment traduire les messages d'erreur (en fait cela ne s'appliquera qu'à l'échantillon de codes qui vient d'être présenté).

Listing 10.70 : error_inc.php

```
<?php

// Propose une version Française de ociError

function ociErrorFr($id="")
{

    if ($id == "") {
        $errEn = oci_error();
    } else {
        $errEn = oci_error($id);
    }

    if ($errEn === FALSE) return FALSE;

    switch ($errEn["code"]) {
        case   900 : $errEn["message"] = "ORA-00900: Requête SQL invalide";
                     break;
        case  1017 : $errEn["message"] =
                          "ORA-01017: Nom utilisateur/Mot de passe invalide:".
                          " connexion refusée";
                     break;
        case 12154 : $errEn["message"] =
                          "ORA-12154: Aucune référence trouvée par le TNS";
                     break;
}
    return $errEn;
}
?>
```

Et voici un script générant l'ensemble des erreurs ci-dessus, afin de bien mettre en évidence le bon fonctionnement de cette fonction personnalisée.

Listing 10.71 : error.php

```
<?php

    // Paramètres du script
    require_once("parametres_bd_inc.php");
    $table = "error01";

    // Chargement de la fonction de traduction des messages d'erreur
    require_once("error_inc.php");

    // Erreur d'identification
    $idConnexion = @oci_pconnect('xyz', 'xyz', $base);
    print_r(ociErrorFr());
    echo "<br />";

    // Base de données inaccessible
    $idConnexion = @oci_pconnect($utilisateur, $motDePasse, "xyz");
    print_r(ociErrorFr());
    echo "<br />";


    // Connexion à la base de données
    $idConnexion = oci_pconnect($utilisateur, $motDePasse, $base);
    if (!$idConnexion) {
        die("<b>Impossible de se connecter à la base de données".
            " [$base] sous le compte [$utilisateur]".
            " avec le mot de passe [$motDePasse].</b>");
    }

    // requête invalide
    $idRequete = @oci_parse($idConnexion, "XYZ");
    @oci_execute($idRequete);

    echo "derniere erreur<br />";
    print_r(ociErrorFr());
    echo "derniere erreur connexion<br />";
    print_r(ociErrorFr($idConnexion));
    echo "derniere erreur requete<br />";
    print_r(ociErrorFr($idRequete));
?>
```

Ce script génère alors une page contenant les messages :

```
Array( [code] => 1017 [message] => ORA-01017: Nom utilisateur/Mot de passe
✂ invalide: connexion refusée )
Array( [code] => 12154 [message] => ORA-12154: Aucune référence trouvée par le
✂ TNS )
```

```
derniere erreur
derniere erreur connexion
derniere erreur requete
Array( [code] => 900 [message] => ORA-00900: Requête SQL invalide )
```


Exemples d'applications

Compteur de clics

Une application courante d'utilisation des bases de données, et simple à mettre en œuvre, consiste à créer un compteur de clics.

En effet, peut-être souhaitez-vous proposer, sur votre site, un certain nombre de liens vers des sites Internet (annuaire web) ou vers des fichiers (bibliothèques de scripts). Peut-être que certains (ou la totalité) de ces liens sont proposés par un partenaire. Bref, tout cela vous incite à savoir quelles sont les pages les plus fréquemment consultées (pour connaître les centres d'intérêt des visiteurs) et combien de visiteurs vous avez redirigés vers votre sponsor.

La méthode la plus classique consiste à stocker en base de données l'ensemble des liens ainsi proposés, et à remplacer les traditionnels liens de la forme **http://www.domaine.com** par un lien vers un script chargé d'incrémenter le compteur correspondant au site indiqué, et de rediriger le client vers le site grâce à la fonction `header()`.

Pour notre exemple, nous utiliserons une table appelée *url* contenant trois champs :

- *urlid*, identifiant de l'URL (champ auto-incrémenté) ;
- *url*, l'URL proprement dite ;
- *nbclic*, le nombre de clics.

Le script de redirection (appelé ici *compteurclic_redirection.php*) devra alors accepter un paramètre (`urlid`) et devra, à partir de ce paramètre, déterminer quelle est la valeur du champ *url*, incrémenter *nbclic* pour cette URL, et rediriger vers *url*.

Les appels à ce script auront alors la forme **http://localhost/compteurclic_redirection.php?urlid=3** (pour accéder à l'URL ayant l'identifiant 3). À supposer que l'URL 3 corresponde au site **http://www.sqlfacil.com**, cela signifie que le lien `<a href="http://www.sqlfacile.com">` devra être remplacé par `<a href="compteurclic_redirection.php?urlid=3>` si l'on souhaite en compter le nombre de clics.

Concrètement, cette table pourra être créée et alimentée avec la fonction `CC_initialiseBD()` du script suivant :

Listing 10.72 : compteurclic_admin_inc.php

```
<?php

// Fonction chargé de créer et d'alimenter
// la table "compteur de clics"

function CC_initialiseBD($idConnexion, $table)
```

```
{

    // Création de la table
    $requete = "DROP TABLE $table";
    $idRequete = oci_parse($idConnexion, $requete);
    @oci_execute($idRequete);

    $requete = "CREATE TABLE $table (urlId INTEGER,".
                                        "url VARCHAR(128) NOT NULL,".
                                        "nbclic INTEGER DEFAULT 0,".
                                        "UNIQUE(url))";
    $idRequete = oci_parse($idConnexion, $requete);
    if (!oci_execute($idRequete)) return FALSE;

    // Création d'une séquence pour les identifiants d'url
    $requete = "DROP SEQUENCE seq_$table";
    $idRequete = oci_parse($idConnexion, $requete);
    @oci_execute($idRequete);

    $requete = "CREATE SEQUENCE seq_$table";
    $idRequete = oci_parse($idConnexion, $requete);
    if (!oci_execute($idRequete)) return FALSE;

    // Création d'un trigger pour remplir automatiquement
    // le champ identifiant d'url à chaque ajout d'une url
    // (compteur)

    $requete = "DROP TRIGGER compteur_$table";
    $idRequete = oci_parse($idConnexion, $requete);
    @oci_execute($idRequete);

    $idRequete = oci_parse($idConnexion,
                    "CREATE TRIGGER compteur_$table".
                    " BEFORE INSERT ON $table ".
                    "FOR EACH ROW ".
                    "BEGIN ".
                    "SELECT seq_$table.NEXTVAL INTO :NEW.urlId FROM DUAL; ".
                    "END;");
    oci_execute($idRequete);

    // Prépare une requête
    $idRequete = oci_parse($idConnexion,
                    "INSERT INTO $table (url) VALUES (:url)");

    // Associe une variable SQL à une variable PHP
    oci_bind_by_name($idRequete, ":url", $url, 129);

    // Alimentation de la base de données
    $url = "http://www.php.net";
    oci_execute($idRequete);

    $url = "http://www.phpfacile.com";
```

```
    oci_execute($idRequete);

    $url = "http://www.sqlfacile.com";
    oci_execute($idRequete);

    $url = "http://www.xmlfacile.com";
    oci_execute($idRequete);

    $url = "http://www.ootoogo.com";
    oci_execute($idRequete);

    return TRUE;
}
?>
```

Les opérations de connexion n'apparaissent pas dans ce script, mais vous pouvez consulter le script *compteurclic_admin.php* si vous le souhaitez.

La lecture de la liste des liens disponibles en base de données pourra se faire via la fonction `CC_recupereLiens()` du script suivant :

Listing 10.73 : compteurclic_inc.php

```
<?php

// Fonction retournant les informations de liens
// sous forme d'un tableau associatif possedants
// les clés
// - "lien" associé au tableau des liens hypertextes
// - "nbclic" associé au tableau des nombres de clics

function CC_recupereLiens($idConnexion, $table)
{
    // Nom du script chargé du comptage et de la redirection
    $script = "compteurclic_redirection.php";

    // Préparation de la requête SELECT
    $idRequete = oci_parse($idConnexion, "SELECT * FROM $table");

    // Exécution de la requête
    oci_execute($idRequete);

    // Récupération des enregistrements les uns après les autres
    while (oci_fetch_into($idRequete, $enreg, OCI_ASSOC)) {
        $liens["lien"][] = "<a href=\"$script?urlid=".$enreg["URLID"]."\">".
                           $enreg["URL"]."</a>";
        $liens["nbclic"][] = $enreg["NBCLIC"];
    }

    return $liens;
}
```

```
?>
```

L'affichage des liens consiste uniquement à mettre en page le contenu du tableau ainsi récupéré (voir la fonction du script *compteurclic_affichage_inc.php*).

Comme cela a été précisé, le lien `<a href="http://www.sqlfacile.com">http://www.sqlfacile.com</a>` a été remplacé par `<a href="compteurclic_redirection.php?urlid=3">http://www.sqlfacile.com </a>`. La redirection vers le site **www.sqlfacile.com** est donc à la charge du script *compteurclic_redirection.php*.

Listing 10.74 : compteurclic_redirection.php

```
<?php

    // Paramètres du script
    require_once("parametres_bd_inc.php");
    $table = "compteurclic";

    // Connexion à la base de données
    $idConnexion = @oci_pconnect($utilisateur, $motDePasse, $base);
    if (!$idConnexion) {
        die("<b>Impossible de se connecter à la base de données".
            " [$base] sous le compte [$utilisateur]".
            " avec le mot de passe [$motDePasse].</b>");
    }

    // Récupération du paramètre urlid passé
    // en paramètre de ce script par la méthode GET
    $urlid = $_GET["urlid"];

    // Recherche dans la base de donnée de l'url
    // correspondant à $urlid
    $idRequete = @oci_parse($idConnexion,
                            "SELECT * FROM $table WHERE urlid=$urlid");
    @oci_execute($idRequete);
    @oci_fetch_into($idRequete, $enreg, OCI_ASSOC) or
      die("<b>Impossible de trouver cette url<b>");

    // Nous connaissons maintenant l'url
    // nous pourrons nous rediriger vers $enreg["URL"];
    // une fois le compteur incrémenté
    $url = $enreg["URL"];
    $idRequete = @oci_parse($idConnexion,
                            "UPDATE $table SET nbclic=nbclic+1".
                            " WHERE urlid=$urlid");
    @oci_execute($idRequete);

    // Dans le cas d'une connexion non persistante
    // (utilisation de oci_connect ou oci_new_connect)
    // il faudrait également la ligne suivante
    // oci_close($idConnexion);
```

```
    // C'est l'heure de la redirection
    header("Location: $url");

    // REM: L'appel à header ne fonctionnera pas si des données
    // ont déjà été émises vers le navigateur. Ce qui peut
    // arriver si l'un des appels précédents affiche un message
    // d'alerte. Il était donc important de faire précéder
    // ces appels par le signe @

    // REM 2: Avec Oracle, nous aurions pu utiliser une procédure stockée
    // réalisant à la fois l'opération de lecture et celle de mise à jour
    // du compteur

?>
```

REMARQUE

Utilisation du bouton retour

Si, après avoir suivi un lien, vous revenez sur la page cliccompte.php en utilisant le bouton retour (back), le contenu de la page ne sera pas réactualisé. Par conséquent, le nombre de visites affiché ne sera pas correct. N'oubliez donc pas, dans ce cas, de rafraîchir (bouton actualiser) la page.

SuperTheque

L'essentiel du code de l'application a été présenté en introduction de ce chapitre. Il ne restait plus qu'a connaître les fonctions Oracle pour implémenter l'interface `RessourceInterface` c'est désormais chose faite:

Listing 10.75 : RessourceOracle_class.php

```
<?php
include_once("RessourceInterface_class.php");
include_once(dirname(__FILE__)."/../config/oracle_cfg.php");
/**
 * RessourceOracle_class.php
 * Classe d'acces a une base de donnees Oracle
 * Cette classe doit implementer toutes les methodes de l'interface
 *   RessourceInterface.
 * Compatibilite: PHP 5
 */
class Ressource implements RessourceInterface
{
    var $idConnexion;

    public function connexion()
    {
        global $oci_serveur, $oci_utilisateur, $oci_motDePasse;
        global $oci_base;
```

```
    if (!isset($this->idConnexion)) {
        $idConnexion = oci_pconnect($oci_utilisateur,
                                    $oci_motDePasse,
                                    $oci_base);
        if (!$idConnexion) return FALSE;
        $this->idConnexion = $idConnexion;
    }
    return TRUE;
}

public function deconnexion()
{
    // Rien a faire, il s'agit d'une connexion persistante
}

public function reset()
{
    $requete = "DROP TABLE types";
    $idRequete = oci_parse($this->idConnexion, $requete);
    @oci_execute($idRequete);
    // Ne pas tenir compte d'une eventuelle erreur
    // la table n'existe peut-etre tout simplement pas

    $requete = "CREATE TABLE types (".
                    "id INTEGER PRIMARY KEY,".
                    "type VARCHAR(255))";
    $idRequete = oci_parse($this->idConnexion, $requete);
    if (!oci_execute($idRequete)) return FALSE;

    $requete = "DROP SEQUENCE seq_types";
    $idRequete = oci_parse($this->idConnexion, $requete);
    @oci_execute($idRequete);

    $requete = "CREATE SEQUENCE seq_types";
    $idRequete = oci_parse($this->idConnexion, $requete);
    if (!oci_execute($idRequete)) return FALSE;

    $requete = "DROP TRIGGER compteur_types";
    $idRequete = oci_parse($this->idConnexion, $requete);
    @oci_execute($idRequete);

    $requete = "CREATE TRIGGER compteur_types ".
                   "BEFORE INSERT ON types ".
                   "FOR EACH ROW ".
                   "BEGIN ".
                   "SELECT seq_types.NEXTVAL INTO :NEW.id FROM DUAL; ".
                   "END;";
    $idRequete = oci_parse($this->idConnexion, $requete);
    if (!oci_execute($idRequete)) return FALSE;


    $types = array("Album", "Film", "Livre", "Musique");
```

```
        foreach ($types as $type) {
            $requete = "INSERT INTO types (type) VALUES ('$type')";
            $idRequete = oci_parse($this->idConnexion, $requete);
            if (!oci_execute($idRequete)) return FALSE;
        }

        $requete = "DROP TABLE articles";
        $idRequete = oci_parse($this->idConnexion, $requete);
        @oci_execute($idRequete);
        // Ne pas tenir compte d'une eventuelle erreur
        // la table n'existe peut-etre tout simplement pas

        $requete = "CREATE TABLE articles (".
                    "id INTEGER PRIMARY KEY,".
                    "albumId INTEGER,".
                    "titre VARCHAR(255),".
                    "typeId INTEGER,".
                    "commentaire VARCHAR(255))";
        $idRequete = oci_parse($this->idConnexion, $requete);
        if (!oci_execute($idRequete)) return FALSE;

        $requete = "DROP SEQUENCE seq_articles";
        $idRequete = oci_parse($this->idConnexion, $requete);
        @oci_execute($idRequete);

        $requete = "CREATE SEQUENCE seq_articles";
        $idRequete = oci_parse($this->idConnexion, $requete);
        if (!oci_execute($idRequete)) return FALSE;

        $requete = "DROP TRIGGER compteur_articles";
        $idRequete = oci_parse($this->idConnexion, $requete);
        @oci_execute($idRequete);

        $requete = "CREATE TRIGGER compteur_articles ".
                       "BEFORE INSERT ON articles ".
                       "FOR EACH ROW ".
                       "BEGIN ".
                       "SELECT seq_articles.NEXTVAL INTO :NEW.id FROM DUAL; ".
                       "END;";
        $idRequete = oci_parse($this->idConnexion, $requete);
        if (!oci_execute($idRequete)) return FALSE;
    }

    public function addArticle($albumId,
                               $titre,
                               $typeId,
                               $commentaire)
    {
        $requeteDebut = "INSERT INTO articles (albumId, titre, typeId";
        $requeteFin = ") VALUES ($albumId,".
                                 "'".sqlite_escape_string($titre)."',".
                                 "$typeId";
```

```
    if ($commentaire != "") {
         $requeteDebut .= ",commentaire";
         $requeteFin .= ",'".sqlite_escape_string($commentaire)."'";
    }
    $requete = $requeteDebut . $requeteFin . ")";
    $idRequete = oci_parse($this->idConnexion, $requete);
    if (!oci_execute($idRequete)) return FALSE;
}

public function getArticle($articleId)
{
    $requete = "SELECT * FROM articles WHERE id=$articleId";
    $idRequete = oci_parse($this->idConnexion, $requete);
    if (!oci_execute($idRequete)) return FALSE;

    $article = NULL;
    if ($enreg = oci_fetch_array($idRequete)) {
        $article = $this->enreg2Article($enreg);
    }
    return $article;
}

public function getArticles(Filtre $filtre, Plage $plage, Tri $tri)
{
    $requete = "SELECT * FROM articles";
    if ($filtre->getAlbumId() != -1) {
        $conditionSQL = "albumId=".$filtre->getAlbumId();
    }
    if ($filtre->getTitre() != "") {
        $conditionSQL = $conditionSQL.
                        " AND titre LIKE '".
                        addSlashes($filtre->getTitre())."'";
    }
    if ($filtre->getTypeId() != -1) {
        $conditionSQL = $conditionSQL.
                        " AND typeId=".$filtre->getTypeId();
    }

    if ($tri->getSens() == -1) {
        $triSQL = "ORDER BY ".$tri->getChamp()." DESC";
    } else if ($tri->getSens() == 1) {
        $triSQL = "ORDER BY ".$tri->getChamp()." ASC";
    }

    $requete = $requete." WHERE ".$conditionSQL." ".
                                  $triSQL;

    $requete = "SELECT * FROM (".
                  "SELECT id, albumId, titre, typeId, commentaire,".
                        " ROWNUM noligne ".
                  "FROM (".$requete.")".
               ") WHERE noligne BETWEEN ".($plage->getPremierArticle()+1).
```

```
                        " AND ".($plage->getPremierArticle()+
                                 $plage->getNbArticleMax());
    $idRequete = oci_parse($this->idConnexion, $requete);
    if (!oci_execute($idRequete)) return FALSE;

    $articles = NULL;
    while ($enreg = oci_fetch_array($idRequete)) {
        $articles[] = $this->enreg2Article($enreg);
    }
    return $articles;
}

public function getNbTotalArticles(Filtre $filtre)
{
    $requete = "SELECT COUNT(*) FROM articles";
    if ($filtre->getAlbumId() != -1) {
        $conditionSQL = "albumId=".$filtre->getAlbumId();
    }
    if ($filtre->getTitre() != "") {
        $conditionSQL = $conditionSQL.
                        " AND titre LIKE '".
                        addSlashes($filtre->getTitre())."'";
    }
    if ($filtre->getTypeId() != -1) {
        $conditionSQL = $conditionSQL.
                        " AND typeId=".$filtre->getTypeId();
    }

    $requete = $requete." WHERE ".$conditionSQL;
    $idRequete = oci_parse($this->idConnexion, $requete);
    if (!oci_execute($idRequete)) return FALSE;

    if ($enreg = oci_fetch_array($idRequete)) {
        return $enreg[0];
    } else return FALSE;
}

public function getTypes()
{
    $requete = "SELECT * FROM types";
    $idRequete = oci_parse($this->idConnexion, $requete);
    if (!oci_execute($idRequete)) return FALSE;

    $types = NULL;
    while ($enreg = oci_fetch_array($idRequete)) {
        $types[$enreg["ID"]] = $enreg["TYPE"];
    }
    return $types;
}

public function getAlbumTypeId()
{
```

```
            return 1;
        }


        private function enreg2Article($enreg)
        {
            // ATTENTION: Avec Oracle les noms des cles
            // du tableau sont en majuscules.
            $article = new Article();
            $article->setId($enreg["ID"]);
            $article->setAlbumId($enreg["ALBUMID"]);
            $article->setTitre($enreg["TITRE"]);
            $article->setTypeId($enreg["TYPEID"]);
            $article->setCommentaire($enreg["COMMENTAIRE"]);
            return $article;
        }
}
?>
```


Comme vous le constatez, la principale difficulté n'est pas tant au niveau des fonctions disponibles qu'au niveau de la construction des requêtes SQL. La requête sera simple lorsqu'il ne s'agit que de récupérer un enregistrement identifié par la clé `Id` (comme c'est le cas avec `getArticle()`) elle sera bien plus complexe s'il s'agit des récupérer un ensemble d'enregistrements répondant à des critères précis et triés (comme c'est le cas avec `getArticles()`).

Requêtes avec des apostrophes

Lors de la construction de la requête, il faut bien garder à l'esprit que l'un des éléments de la requête (typiquement un champ de type texte saisi par l'utilisateur, comme ici le titre ou le commentaire) peut contenir une apostrophe qui pourrait être confondu avec le délimiteur de chaîne. Pensez donc bien à faire un appel à `addSlashes()` lorsque cela peut s'avérer nécessaire.

Affichage par page

Des bases de données telles que MySQL proposent une instruction `LIMIT` permettant de ne retourner qu'un sous-ensemble des résultats de la requête, en précisant l'index du premier enregistrement et le nombre de résultats à retourner. Il n'y a malheureusement pas d'équivalent sous Oracle ; il faut donc avoir recours à quelques acrobaties pour obtenir le même résultat.

Ainsi, pour retourner les vingt premiers enregistrements à partir du onzième de la requête `<requete>`, il faut exécuter la requête :

```
SELECT * FROM (SELECT champ1, champ2, ..., ROWNUM noligne FROM (<requete>))
✂ WHERE noligne BETWEEN 11 AND 30)
```

Tri

Modifier l'ordre d'affichage des annonces n'est pas non plus bien sorcier. Le tri peut s'effectuer directement au niveau de la requête SQL grâce à l'instruction `ORDER BY`.

Moteur de recherche (filtre)

Pour filtrer, il suffit d'utiliser l'instruction `WHERE`. Dans notre cas, nous souhaitons autoriser l'utilisation de jokers (comme %) afin, par exemple, de rechercher les titres commençant par une chaîne donnée. C'est pourquoi, nous ne ferons pas un test de type `titre='motcle'` mais `titre LIKE 'motcle'` (où `motcle` pourrait avoir la valeur "La 7eme compagnie%").

En savoir plus…

... sur le résultat d'une requête

Il vous est ainsi possible de connaître le nombre de champs retournés par une requête (et, par conséquent, le nombre de champs d'une table dans le cas d'une requête `SELECT * FROM <nom_table>` sur une table non vide).

oci_num_fields() (ociNumCols() PHP4)

Retourne le nombre de champs dans les enregistrements retournés par une requête SQL.

Syntaxe	`int oci_num_fields(resource $idRequete)`
`$idRequete`	Identifiant de requête tel que retourné par `oci_execute()`.
`retour`	Nombre de champs.

Mais aussi, le nom de ces champs :

oci_field_name() (ociColumnName() PHP4)

Retourne le nom des champs dans les enregistrements retournés par une requête SQL.

Syntaxe	`string oci_field_name(resource $idRequete, int $index)`
`$idRequete`	Identifiant de requête tel que retourné par `oci_execute()`.
`$index`	Index du champ dont vous souhaitez connaître le nom. Le nombre total de champs peut être obtenu en faisant appel à `oci_num_fields()`.ATTENTION : l'indexation commence à 1.
`retour`	Nom du champ.

Cependant, les champs ont d'autres informations à nous livrer que leur nom. Pour cela, vous disposez des fonctions :

oci_field_type() (ociColumnType() PHP4)

Retourne le type des champs dans les enregistrements retournés par une requête SQL.

Syntaxe	`string oci_field_type(resource $idRequete, int $index)`
`$idRequete`	Identifiant de requête tel que retourné par `oci_execute()`.
`$index`	Index du champ dont vous souhaitez connaître le type. Le nombre total de champs peut être obtenu en faisant appel à `oci_num_fields()`.ATTENTION : l'indexation commence à 1.
`retour`	Type du champ, ou `FALSE` en cas d'erreur.

oci_field_size() (ociColumnSize() PHP4)

Retourne la taille (en octets) des champs de la table.

Syntaxe	`int oci_field_size(resource $idRequete, int $index)`
`$idRequete`	Identifiant de requête tel que retourné par `oci_execute()`.
`$index`	Index du champ dont on souhaite connaître la taille. Le nombre total de champs peut être obtenu en faisant appel à `oci_num_fields()`. ATTENTION : l'indexation commence à 1.
`retour`	Taille (en octets) occupée par le champ, ou `FALSE` en cas d'erreur.

Listing 10.76 : info01_inc.php

```
<?php

// Fonction affichant certaines informations liées
// à la structure d'une table

function describe($idConnexion, $table)
{
    // Préparation de la requête SELECT
    $idRequete = oci_parse($idConnexion, "SELECT * FROM $table");

    // Exécution de la requête
    oci_execute($idRequete);

    // Récupération du nombre de champs retournés
    $nbChamps = oci_num_fields($idRequete);

    // Pour chaque champ, récupération des informations
    // - nom du champ
    // - type du champ
    // - taille du champ
    for ($i=1; $i<=$nbChamps; $i++) {
```

```
        echo "<b>".oci_field_name($idRequete, $i)."</b>";
        echo " de type <i>".oci_field_type($idRequete, $i)."</i>";
        echo " de longueur <i>".oci_field_size($idRequete, $i)."</i><br />";
    }
}
?>
```


Les numéros de version

La fonction évoquée ici n'est pas d'une utilisation très courante, mais pourra éventuellement vous servir si vous êtes amené à utiliser des instructions avancées de la base de données Oracle (instructions implémentées dans des versions récentes, par exemple).

oci_server_version() (ociServerVersion() PHP4)

Retourne le numéro de version du serveur Oracle.

Syntaxe	`string oci_serveur_version(resource $idBaseDonnees)`
`$idBaseDonnees`	Identifiant de connexion à une base de données tel que retourné par `oci_connect()`, `oci_new_connect()` ou `oci_pconnect()`.
`retour`	Numéro de version du serveur.

La chaîne retournée sera de la forme :

```
Oracle Database 10g Release 10.1.0.2.0 - Production
```

10.12. SQLite

A la différence des autres bases de données présentées, SQLite est totalement intégrée à PHP 5.

Installation

Comme SQLite est intégrée à PHP 5, la procédure d'installation est extrêmement facile vu qu'il n'y a rien à faire.

Installer SQLite pour PHP 4 (ou 3)
Sous windows, il vous suffit de récupérer le fichier php_sqlite.dll disponible sur le cédérom et de charger la librairie.

Utilisation

Comme indiqué dans le chapitre d'introduction, l'utilisation basique de la base de données s'opère selon le schéma suivant :

- Connexion grâce aux fonctions `sqlite_open()` ou `sqlite_popen()`.
- Soumission de la requête via la fonction `sqlite_query()`, `sqlite_array_query()` ou `sqlite_unbuffered_query()`.
- Déconnexion grâce à la fonction `sqlite_close()`.

Connexion

Comme pour toute base de données avant même de pouvoir effectuer une requête, une connexion doit être établie avec la base de données.

sqlite_open()

Établit une connexion (non persistante) avec le serveur de bases de données. La base de données est crée si elle n'existe pas déjà.

Syntaxe	`resource sqlite_open(string $nomFichier [, int $mode [, string &$messageErreur]])`
`$nomFichier`	Nom du fichier représentant la base de données. Si le fichier n'existe pas il est crée (et la base de données associée également). Le chemin peut être relatif ou absolu, assurer vous d'avoir l'autorisation en écriture dans le répertoire où vous voulez créer le fichier.
`$mode`	A ce jour cet argument est ignoré, il est censé définir le mode d'accés a la base de données. La syntaxe est celle du monde Unix (cf. chmod) par défaut sa valeur est 0666 (lecture et écriture pour tous), on pourrait autoriser la lecture seule pour tous par la valeur 0444.
`&$messageErreur`	Message d'erreur expliquant pourquoi la base de données n'a pu être ouverte.
`retour`	Identifiant de connexion au serveur de bases de données, ou `FALSE` en cas d'échec.

Base de données en mémoire

Vous pouvez créer une base de données en mémoire en remplaçant le nom du fichier par :memory: mais attention cela ne peut servir que pour une base de données temporaire vu qu'elle sera effacée dés la fin du script en cours.

L'utilisation d'une connexion persistante requiert, à la place, l'appel à la fonction `sqlite_popen()`.

sqlite_popen()

Établit une connexion persistante avec le serveur de bases de données. Si une connexion persistante est disponible, elle sera utilisée ; sinon, une nouvelle connexion persistante sera créée.

Syntaxe	`resource sqlite_popen(string $nomFichier [, int $mode [, string &$messageErreur]])`
`$nomFichier`	Nom du fichier représentant la base de données. Si le fichier n'existe pas il est crée (et la base de données associée également). Le chemin peut être relatif ou absolu, assurer vous d'avoir l'autorisation en écriture dans le répertoire où vous voulez créer le fichier.
`$mode`	A ce jour cet argument est ignoré, il est censé définir le mode d'accés a la base de données. La syntaxe est celle du monde Unix (cf. chmod) par défaut sa valeur est 0666 (lecture et écriture pour tous), on pourrait autoriser la lecture seule pour tous par la valeur 0444.
`&$messageErreur`	Message d'erreur expliquant pourquoi la base de données n'a pu être ouverte.
`retour`	Identifiant de connexion au serveur de bases de données, ou `FALSE` en cas d'échec.

Exécution de la requête SQL

Pour exécuter une requête SQL, il y a différentes méthode en fonction de l'utilisation du résultat. Pour une requête qui retourne un nombre conséquant de lignes (plus de 45), l'utilisation de `sqlite_unbuffered_query()` est conseillée et l'est encore plus si l'on veut parcourir une à une les lignes du résultat. La fonction `sqlite_query()` quand à elle lance une requête sur la base de données et retourne un résultat plus manipulable que `sqlite_unbuffered_query()` car l'ensemble des lignes est mis en mémoire, on peut en connaître le nombre exact ou faire des marches arrières.

sqlite_query()

Exécute une requête SQL. Cette fonction est idéale pour faire des mises à jour de tables mais pour un SELECT retournant de nombreux résultats, la fonction `sqlite_unbuffered_query()` est plus appropriée.

Syntaxe	`resource sqlite_query((string $requete, resource $idConnexion)` `resource sqlite_query(resource $idConnexion, string $requete)`
`$requete`	Requête SQL.
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `sqlite_open()` ou `sqlite_popen()`.
`retour`	Identifiant de requête SQL, ou `FALSE` en cas d'échec.

sqlite_unbuffered_query()

Exécute une requête SQL sans mise en mémoire du résultat. Cette fonction est particulièrement adaptées pour un parcours unique d'un ensemble de lignes.

Syntaxe	`resource sqlite_unbuffered_query(string $requete, resource $idConnexion)`
	
	`resource sqlite_unbuffered_query(resource $idConnexion, string $requete)`

`$requete`	Requête SQL.
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `sqlite_open()` ou `sqlite_popen()`.
`retour`	Identifiant de requête SQL, ou `FALSE` en cas d'échec.

Vous serrez certainement ammenés à vouloir plus de flexibilité et pouvoir accéder à differentes lignes du résultat dans un ordre différent de l'ordre de sortie. Pour cela vous mettrez probablement les données dans un tableau. La fonction `sqlite_array_query()` fait exactement ce travail pour vous mais de façon plus rapide, elle retourne un tableau des lignes du résultat. Mais attention, ces opérations sont extremement coûteuses en mémoire, il faut donc utiliser ces méthodes que très ponctuellement pour des résultats de moins de 45 lignes environ.

sqlite_array_query()

Exécute une requête SQL et stocke le résultat dans un tableau.

Syntaxe	`resource sqlite_array_query(string $requete, resource $idConnexion [, int $typeResultat [, boolean $decodeBinaire]])`
	
	`resource sqlite_array_query(resource $idConnexion, string $requete [, int $typeResultat [, bool $decodeBinaire]])`

`$requete`	Requête SQL.
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `sqlite_open()` ou `sqlite_popen()`.
`$typeResultat`	Par défaut cette valeur est mise a `SQLITE_BOTH` qui signifie que les indices du tableau seront le numéro et le nom de colonne. Pour n'avoir que le numéro de colonne il suffit de mettre `SQLITE_NUM` et pour le nom de colonne, `SQLITE_ASSOC`.
`$decodeBinaire`	Par défaut (`TRUE`) PHP décodera le résultat binaire si il a été encodé par `sqlite_escape_string()`.
`retour`	Identifiant de requête SQL, ou `FALSE` en cas d'échec.

Dans le cas d'une requête retournant un résultat (typiquement une requête SELECT), il faudra également analyser le résultat. Mais nous verrons cela un peu plus loin.

Déconnexion

Pour vous déconnecter – opération théoriquement facultative mais toutefois vivement conseillée –, vous disposez de la fonction `sqlite_close()`.

sqlite_close

Met fin à la connexion à la base de données et libère les ressources associées.

Syntaxe	`void sqllite_close(resource $idConnexion)`
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `sqlite_open()` ou `sqlite_popen()`. Si la connexion était persistante, elle est fermée et retirée de la liste des connexions persistantes.

Lecture des enregistrements

Dans ce premier exemple nous allons juste nous connecter à la base de données, créer une table, y insérer une donnée et la retrouver par trois requêtes SQL successives.

Nous avons vu les fonctions d'ouverture, de fermeture et de requête, voyons une fonction qui nous manque pour le cours exemple qui suit.

sqlite_fetch_array()

Retourne la ligne courante du résultat d'une requête sous la forme d'un tableau, le curseur passe à la ligne suivante une fois la fonction terminée. Les éléments pourront être récupérés soit par leur indice de colonne, soit par le nom de colonne.

Syntaxe	`array sqlite_fetch_array(resource $idRequete [, int $typeResultat [, booleanl $decodeBinaire]])`
`$idRequete`	Identifiant de la requête.
`$typeResultat`	Par défaut cette valeur est mise a `SQLITE_BOTH` qui signifie que les indices du tableau seront le numéro et le nom de colonne. Pour n'avoir que le numéro de colonne il suffit de mettre `SQLITE_NUM` et pour le nom de colonne, `SQLITE_ASSOC`.
`$decodeBinaire`	Par défaut (`TRUE`) PHP décodera le résultat binaire si il a été encodé par `sqlite_escape_string()`.
`retour`	Un tableau représentant la ligne suivante du resultat en cours de lecture.

Listing 10.77 : sqlite_open.php

```
<?php
 // Ouverture et creation de la base de donnees maBaseDeDonnees
 $bd = sqlite_open('maBaseDeDonnees', 0666, $sqliteerror) or die($sqliteerror);
 // Creation de la table maTable caracterisee par deux colonnes, maColonne1
 ✂ d'entier
 // et maColonne2 de chaines d'au plus 25 caracteres
 sqlite_query($bd, 'CREATE TABLE maTable (maColonne1 int, maColonne2
 ✂ varchar(16))');
 // Insertion de donnees
 sqlite_query($bd, "INSERT INTO maTable VALUES (12, 'toto')");
 // Recherche des donnees
 $result = sqlite_query($bd, 'select * from maTable');
 // Recuperation de la premiere ligne de resultat
 $ligne = sqlite_fetch_array($result);
 // Affichage de la ligne par les indices de colonnes
 echo($ligne[0].":".$ligne[1]."<br />\n");
 // Affichage de la ligne par les noms de colonnes
 echo($ligne['maColonne1'].":".$ligne['maColonne2']."<br />\n");
 sqlite_close($bd);
?>
```

La fonction `sqlite_current()` très similaire a `sqlite_fetch_array()` permet de mettre une ligne dans un tableau sans passer à la ligne suivante. Cette fonction ne fonctionnera pas si vous avez utilisé `sqlite_unbuffered_query()`.

sqlite_current()

Retourne la ligne courante du résultat d'une requête sous la forme d'un tableau sans déplacer le curseur. Les éléments pourront être récupérés soit par leur indice de colonne, soit par le nom de colonne.

Syntaxe	`array sqlite_current(resource $idRequete [, int $typeResultat [, boolean $decodeBinaire]])`
`$idRequete`	Identifiant de la requête.
`$typeResultat`	Par défaut cette valeur est mise a `SQLITE_BOTH` qui signifie que les indices du tableau seront le numéro et le nom de colonne. Pour n'avoir que le numéro de colonne il suffit de mettre `SQLITE_NUM` et pour le nom de colonne, `SQLITE_ASSOC`.
`$decodeBinaire`	Par défaut (`TRUE`) PHP décodera le résultat binaire si il a été encodé par `sqlite_escape_string()`.
`retour`	Un tableau représentant la ligne suivante du resultat en cours de lecture.

Pour ne pas avoir créer un tableau lorsque l'on veux récuperer qu'un seul champ d'une table (par exemple l'identifiant lors d'une requête du type `'SELECT id FROM maTable'`, la fonction

sqlite_fetch_single() ou sqlite_fetch_string() qui est un alias, permet de récupérer directement le premier champ et de passer à la ligne suivante.

sqlite_fetch_single(), sqlite_fetch_string()

Permet de récupérer directement la valeur du premier champ d'un résultat.

Syntaxe	`string sqlite_fetch_single(resource $idRequete [, boolean $decodeBinaire]])`
`$idRequete`	Identifiant de la requête.
`$decodeBinaire`	Par défaut (`TRUE`) PHP décodera le résultat binaire si il a été encodé par `sqlite_escape_string()`.
`retour`	La valeur du premier champ de la ligne en cours.

Listing 10.78 : sqlite_fetch_single.php

```
<?php
 // Ouverture et creation de la base de donnees test
 $bd = sqlite_open('test', 0666, $sqliteerror) or die($sqliteerror);
 // Creation de la table maTable
 sqlite_query($bd, 'CREATE TABLE maTable
                    (maColonne1 varchar(25), maColonne2 varchar(16))');
 // Insertion de donnees
 sqlite_query($bd, "INSERT INTO maTable VALUES ('Laurent', 'Guedon')");
 sqlite_query($bd, "INSERT INTO maTable VALUES ('Pierre-Emmanuel', 'Muller')");
 sqlite_query($bd, "INSERT INTO maTable VALUES ('Damien', 'Heute')");
 sqlite_query($bd, "INSERT INTO maTable VALUES ('Thomas', 'Heute')");
 // Recherche des donnees
 $result = sqlite_unbuffered_query($bd, 'select * from maTable');
 echo sqlite_fetch_single($result, SQLITE_NUM);
 echo sqlite_fetch_single($result, SQLITE_NUM);
 echo sqlite_fetch_single($result);
 echo sqlite_fetch_single($result);
 sqlite_close($bd);
?>
```

Le résultat attendu est le suivant : (au formattage de texte près)

```
Laurent
Pierre-Emmanuel
Damien
Thomas
```

Si une ligne possède de nombreuses colonnes ou est particulièrement volumineuse (avec des champs binaires par exemple) il est avantageux de ne pas charger toute la ligne mais seulement la colonne intéresssante.

sqlite_column()

Retourne le contenu de la colonne de la ligne courante.

Syntaxe	`mixed sqlite_column(resource $idRequete, mixed $indiceOuNom [, boolean $decodeBinaire])`
`$idRequete`	Identifiant de requête.
`$indiceOuNom`	Numéro de colonne ou son nom.
`$decodeBinaire`	Par défaut (`TRUE`) PHP décodera le résultat binaire si il a été encodé par `sqlite_escape_string()`.
`retour`	Le contenu de la colonne demandée.

Pour parcourir un ensemble de résultats, la fonction `sqlite_has_more()` permet de savoir s'il reste des lignes en attente d'être traitées.

sqlite_has_more()

Cette fonction permet lors du parcours d'un réultat à savoir si il reste des lignes à parcourir.

Syntaxe	`boolean sqlite_has_more(resource $idRequete)`
`$idRequete`	Identifiant de requête.
`retour`	`TRUE` si il reste des lignes a analyser.

Listing 10.79 : sqlite_unbuffered_query.php

```
<?php
 // Ouverture et creation de la base de donnees test
 $bd = sqlite_open('test', 0666, $sqliteerror) or die($sqliteerror);
 // Creation de la table maTable
 sqlite_query($bd, 'CREATE TABLE maTable
                    (maColonne1 varchar(25), maColonne2 varchar(16))');
 // Insertion de donnees
 sqlite_query($bd, "INSERT INTO maTable VALUES ('Laurent', 'Guedon')");
 sqlite_query($bd, "INSERT INTO maTable VALUES ('Pierre-Emmanuel', 'Muller')");
 sqlite_query($bd, "INSERT INTO maTable VALUES ('Damien', 'Heute')");
 sqlite_query($bd, "INSERT INTO maTable VALUES ('Thomas', 'Heute')");
 // Recherche des donnees
 $result = sqlite_unbuffered_query($bd, 'select * from maTable');
 // Recuperation une a une des lignes de resultat
 while (sqlite_has_more($result)) {
   $ligne = sqlite_fetch_array($result, SQLITE_NUM);
   echo($ligne[0]." ".$ligne[1]."<br />\n");
 }
 sqlite_close($bd);
?>
```

Déplacement du curseur

Le parcours d'une table s'effectue avec un curseur. La plupart des fonctions de lecture de lignes déplacent le curseur à la ligne suivante. Il peut-être intéressant de déplacer le curseur de manière plus contrôlée.

Pour déplacer le curseur sur la ligne suivante, il suffit de faire appel à `sqlite_next()`.

sqlite_next()

Cette fonction permet de déplacer le curseur à la ligne suivante. Elle ne fonctionnera pas si vous avez utilisé sqlite_unbuffered_query().

Syntaxe	`boolean sqlite_next(resource $idRequete)`
`$idRequete`	Identifiant de requête.
`retour`	`TRUE` si il y avait une ligne suivante, `FALSE` si c'etait la dernière ligne.

sqlite_rewind()

Cette fonction permet de rammener le curseur à la première ligne. Elle ne fonctionnera pas si vous avez utilisé `sqlite_unbuffered_query()`.

Syntaxe	`boolean sqlite_rewind(resource $idRequete)`
`$idRequete`	Identifiant de requête.
`retour`	`FALSE` si le résultat de la requête est vide.

sqlite_seek()

Cette fonction permet d'ammener le curseur à la ligne voulue. Elle ne fonctionnera pas si vous avez utilisé `sqlite_unbuffered_query()`.

Syntaxe	`boolean sqlite_seek(resource $idRequete, int $numeroLigne)`
`$idRequete`	Identifiant de requête.
`$numeroLigne`	Numéro de la ligne où ammener le curseur.
`retour`	`FALSE` si la ligne n'existe pas, `TRUE` sinon.

Gestion des erreurs

Lors de l'ouverture d'une base de données l'eventuel message d'erreur peut-être récupéré en passant une chaîne de caractère par référence. Deux autres fonctions servent à la gestion des

erreurs, la première permet de récupérer le code d'erreur de la dernière opération effectuée sur une connexion.

sqlite_last_error()

Retourne le code d'erreur de la dernière opération effectuée sur une connexion donnée. (0 si il n'y a pas eu d'erreur)

Syntaxe	`resource sqlite_last_error(resource $idConnexion)`
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `sqlite_open()` ou `sqlite_popen()`.
`retour`	Code d'erreur.

A partir de ce code, il est possible d'avoir une explication textuelle grâce à sqlite_error_string().

sqlite_error_string()

Retourne le message d'erreur correspondant au code fourni.

Syntaxe	`string sqlite_error_string(int $codeErreur)`
`$codeErreur`	Code d'erreur tel que retourné par `sqlite_last_error()`.
`retour`	Message d'erreur.

Appliquer une fonction automatiquement

Particularité de SQLite, il est possible d'appliquer une fonction PHP sur le résultat au niveau de la requête. Avant même de voir la description de `sqlite_create_function()`, voici un petit exemple l'utilisant.

Le programme suivant ouvre une connexion avec une base de données en mémoire, crée une table et y insère des données. Ensuite y est défini la fonction nomComplet() qui concatène deux paramètres séparé d'un espace. L'appel a `sqlite_create_function()` permet de déclarer à SQLite une fonction nom() liée à nomComplet() et qui prend deux arguments en paramètre. La requête est particulière, on applique la fonction nom() sur les deux colonnes ce qui a pour conséquence de n'avoir plus qu'une colonne résultat et ainsi il suffit de faire appel à `sqlite_fetch_single()` pour récupérer le résultat voulu.

Listing 10.80 : sqlite_create_function.php

```
<?php
 // Ouverture et creation de la base de donnees
 $bd = sqlite_open(':memory:', 0666, $sqliteerror) or die($sqliteerror);
 // Creation de la table maTable
```

```
 sqlite_query($bd, 'CREATE TABLE maTable
                    (maColonne1 varchar(25), maColonne2 varchar(16))');

// Insertion de donnees
 sqlite_query($bd, "INSERT INTO maTable VALUES ('Laurent', 'Guedon')");
 sqlite_query($bd, "INSERT INTO maTable VALUES ('Pierre-Emmanuel', 'Muller')");
 sqlite_query($bd, "INSERT INTO maTable VALUES ('Damien', 'Heute')");
 sqlite_query($bd, "INSERT INTO maTable VALUES ('Thomas', 'Heute')");

// Fonction de creation du nom complet
 function nomComplet($prenom, $nom) {
   return $prenom." ".$nom;
 }

 // Declaration a SQLite de la fonction nomComplet
 sqlite_create_function($bd, 'nom', 'nomComplet', 2);

 // Recherche des donnees en appliquant la fonction 'nom'
 $result = sqlite_unbuffered_query($bd, 'select nom(maColonne1, maColonne2) from
 ✂ maTable');

 // Recuperation une a une des lignes de resultat
 while (sqlite_has_more($result)) {
   echo sqlite_fetch_single($result, SQLITE_NUM)."<br />\n";
 }
 sqlite_close($bd);
?>
```

Le résultat attendu est le suivant :

```
Laurent Guedon
Pierre-Emmanuel Muller
Damien Heute
Thomas Heute
```

sqlite_create_function()

Permet de déclarer une fonction appelée directement dans la requête SQL.

Syntaxe	`void sqlite_create_function(resource $idConnexion, string $nomFonction, string mixed $fonction [, int $nombreArguments])`
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `sqlite_open()` ou `sqlite_popen()`.
`$nomFonction`	Nom de la fonction à laquelle on veut associer la fonction PHP.
`$fonction`	Nom de la fonction que vous avez défini précédement.
`$nombreArguments`	Nombre d'arguments que prend la fonction.

En plus de traiter les lignes, il est possible de définir une fonction qui aura pour objectif de traiter l'ensemble des lignes.

sqlite_create_aggregate()

Permet de déclarer une fonction traitant l'ensemble des lignes du résultat.

Syntaxe	`void sqlite_create_aggregate(resource $idConnexion, string $nomFonction, string mixed $fonction, mixed $fonctionFinale [, int $nombreArguments])`
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `sqlite_open()` ou `sqlite_popen()`.
`$nomFonction`	Nom de la fonction à laquelle on veut associer la fonction PHP.
`$fonction`	Nom de la fonction que vous avez défini précédement.
`$fonctionFimale`	Nom de la fonction par laquelle passer l'ensemble du réultat.
`$nombreArguments`	Nombre d'arguments que prend la fonction.

Voici un exemple qui ne sert à rien d'autre que de mieux comprendre :

Listing 10.81 : sqlite_create_aggregate.php

```
<?php
 // Ouverture et creation de la base de donnees test
 $bd = sqlite_open(':memory:', 0666, $sqliteerror) or die($sqliteerror);
 // Creation de la table maTable
 sqlite_query($bd, 'CREATE TABLE maTable
                    (maColonne1 varchar(25), maColonne2 varchar(16))');



 // Fonction de creation du nom complet
 function nomComplet(&$cumul, $prenom, $nom) {
    $cumul[] = $prenom." ".$nom;
 }

 function cumulNom(&$cumul) {
   return $cumul[0].":".$cumul[1].":".$cumul[2].":".$cumul[3];
 }


 // Declaration a SQLite de la fonction nomComplet
 sqlite_create_aggregate($bd, 'nom', 'nomComplet', 'cumulNom', 2);

 // Insertion de donnees
 sqlite_query($bd, "INSERT INTO maTable VALUES ('Laurent', 'Guedon')");
 sqlite_query($bd, "INSERT INTO maTable VALUES ('Pierre-Emmanuel', 'Muller')");
```

```
 sqlite_query($bd, "INSERT INTO maTable VALUES ('Damien', 'Heute')");
 sqlite_query($bd, "INSERT INTO maTable VALUES ('Thomas', 'Heute')");

 // Recherche des donnees en appliquant la fonction 'nom'
 $result = sqlite_unbuffered_query($bd, 'select nom(maColonne1, maColonne2) from
 ✂ maTable');

 echo sqlite_fetch_single($result, SQLITE_NUM)."<br />\n";
 sqlite_close($bd);
?>
```

Le résultat obtenu est le suivant :

```
Laurent Guedon:Pierre-Emmanuel Muller:Damien Heute:Thomas Heute
```

Quelques explications peuvent être utiles. Chacune des entrées est soumise à la fonction `nomComplet()` en plus d'une référence vers une variable (`$cumul` dans l'exemple). Cette dernière permet de garder un pointeur vers un objet (ici un simple tableau). À la fin du traitement de chacune des lignes ce même pointeur est fourni à la fonction `cumulNom()` qui est chargée d'effectuer les dernières opérations sur l'objet. Bien sur dans l'exemple on aurait pu faire bien plus simple, mais ça n'aurait pas été drôle...

Exemples d'applications

Compteur de clics

Une application courante d'utilisation des bases de données, et simple à mettre en œuvre, consiste à créer un compteur de clics.

En effet, peut-être souhaitez-vous proposer, sur votre site, un certain nombre de liens vers des sites Internet (annuaire web) ou vers des fichiers (bibliothèques de scripts). Peut-être que certains (ou la totalité) de ces liens sont proposés par un partenaire. Bref, tout cela vous incite à savoir quelles sont les pages les plus fréquemment consultées (pour connaître les centres d'intérêt des visiteurs) et combien de visiteurs vous avez redirigés vers votre sponsor.

La méthode la plus classique consiste à stocker en base de données l'ensemble des liens ainsi proposés, et à remplacer les traditionnels liens de la forme **http://www.domaine.com** par un lien vers un script chargé d'incrémenter le compteur correspondant au site indiqué, et de rediriger le client vers le site grâce à la fonction `header()`.

Vous pouvez vous reporter à l'annexe "Les en-têtes" pour plus de renseignements sur la redirection.

Pour notre exemple, nous utiliserons une table appelée *url* contenant trois champs :

- *urlid*, identifiant de l'URL (champ auto-incrémenté) ;
- *url*, l'URL proprement dite ;

- *nbclic*, le nombre de clics.

Le script de redirection (appelé ici *compteurclic_redirection.php*) devra alors accepter un paramètre (`urlid`) et devra, à partir de ce paramètre, déterminer quelle est la valeur du champ *url*, incrémenter *nbclic* pour cette URL, et rediriger vers *url*.

Les appels à ce script auront alors la forme **http://localhost/compteurclic_redirection.php?urlid=3** (pour accéder à l'URL ayant l'identifiant 3). À supposer que l'URL 3 corresponde au site **http://www.sqlfacile.com** cela signifie que le lien `<a href="http://www.sqlfacile.com ">` devra être remplacé par `<a href="compteurclic_redirection.php?urlid=3>` si l'on souhaite en compter le nombre de clics.

Concrètement, cette table pourra être créée et alimentée avec la fonction `CC_initialiseBD()` du script suivant :

Listing 10.82 : compteurclic_bd_inc.php (début)

```
<?php
    /**
     * Fonction de connexion à une base de données
     *
     * @return resource Identifiant de connexion
     */
    function CC_connexion()
    {
        $idConnexion = sqlite_open('BaseDeDonneesCompteurClics', 0666,
        ✂ $sqliteerror);
        if (!$idConnexion) return FALSE;
        return $idConnexion;
    }

    /**
     * Fonction de deconnexion.
     * (Ne fait rien sachant que la fonction
     * de connexion établit une connexion persistante)
     */
    function CC_deconnexion()
    {
        return TRUE;
    }

    /**
     * Fonction chargé de créer et d'alimenter
     * la table "compteur de clics"
     **/
    function CC_initialiseBD($idConnexion, $table)
    {
        // Création de la table
        $requete = "DROP TABLE $table";
        @sqlite_query($idConnexion, $requete);

        $requete = "CREATE TABLE $table (".
```

```
                                    "urlId INTEGER ".
                                          "AUTO_INCREMENT PRIMARY KEY,".
                                    "url VARCHAR(128) NOT NULL,".
                                    "nbclic INTEGER DEFAULT 0,".
                                    "UNIQUE(url))";
        if (!sqlite_query($idConnexion, $requete)) return FALSE;

        // Alimentation de la table
        $requete = "INSERT INTO $table (url)".
                        " VALUES ('http://www.php.net')";
        if (!sqlite_query($idConnexion, $requete)) return FALSE;

        $requete = "INSERT INTO $table (url)".
                        " VALUES ('http://www.phpfacile.com')";
        if (!sqlite_query($idConnexion, $requete)) return FALSE;

        $requete = "INSERT INTO $table (url)".
                        " VALUES ('http://www.sqlfacile.com')";
        if (!sqlite_query($idConnexion, $requete)) return FALSE;

        $requete = "INSERT INTO $table (url)".
                        " VALUES ('http://www.xmlfacile.com')";
        if (!sqlite_query($idConnexion, $requete)) return FALSE;

        $requete = "INSERT INTO $table (url)".
                        " VALUES ('http://www.ootoogo.com')";
        if (!sqlite_query($idConnexion, $requete)) return FALSE;

        return TRUE;
    }
?>
```

La lecture de la liste des liens disponibles en base de données pourra se faire via la fonction `CC_recupereLiens()` du script suivant :

Listing 10.83 : compteurclic_bd_inc.php (milieu)

```
<?php
    /**
     * Fonction retournant les informations de liens
     * sous forme d'un tableau associatif possedants
     * les clés
     * - "lien" associé au tableau des liens hypertextes
     * - "nbclic" associé au tableau des nombres de clics
     **/
    function CC_recupereLiens($idConnexion, $table)
    {
        // Nom du script chargé du comptage et de la redirection
        $script = "compteurclic_redirection.php";

        // Requête SELECT
        $requete = "SELECT * FROM $table";
```

```
        $idResultat = sqlite_query($idConnexion, $requete);

        // Récupération des enregistrements les uns après les autres
        while ($enreg = sqlite_fetch_array($idResultat)) {
            $liens["lien"][] = "<a href=\"$script?urlid=".
                               $enreg["urlId"]."\">".
                               $enreg["url"]."</a>";
            $liens["nbclic"][] = $enreg["nbclic"];
        }

        return $liens;
    }
```

L'affichage des liens consiste uniquement à mettre en page le contenu du tableau ainsi récupéré (voir la fonction du script *compteurclic_html_inc.php*).

Comme cela a été précisé, le lien `<a href="http://www.sqlfacile.com">http://www.sqlfacile.com</a>` a été remplacé par `<a href="compteurclic_redirection.php?urlid=3">http://www.sqlfacile.com </a>`. La redirection vers le site **www.sqlfacile.com** est donc à la charge du script *compteurclic_redirection.php*.

Listing 10.84 : compteurclic_redirection.php

```
<?php

    // Paramètres du script
    require_once("compteurclic_bd_inc.php");
    $table = "compteurclic";

    // Connexion à la base de données
    $idConnexion = @CC_connexion();
    if (!$idConnexion) {
        die("<b>Impossible de se connecter à la base de données</b>");
    }

    // Récupération de l'url et incrémentation du compteur
    // pour l'url d'indentifiant passé en paramètre de ce script
    // par la méthode GET
    $url = @CC_recupereUrl($idConnexion, $_GET["urlid"]);

    // Deconnexion
    @CC_deconnexion();

    // C'est l'heure de la redirection
    if ($url) {
        header("Location: $url");
    } else {
        echo "Désolé, nous ne pouvons vous proposer ce lien";
    }?>
```

Ce script appelle principalement la fonction `CC_recupereUrl()` suivante :

Listing 10.85 : compteurclic_bd_inc.php (fin)

```
<?php
    /**
     * Fonction récupérant une url à partir de son identifiant
     * et incrémentant le compteur de clics
     **/
    function CC_recupereUrl($idConnexion, $urlid)
    {
        global $table;

        // Récupère l'url
        $requete = "SELECT * FROM $table WHERE urlid=$urlid";
        $idResultat = sqlite_query($idConnexion, $requete);

        if ($enreg = sqlite_fetch_array($idResultat)) {
            $url = $enreg["url"];

            // Incrémente le compteur
            $requete = "UPDATE $table SET nbclic=nbclic+1 WHERE urlid=$urlid";
            sqlite_query($idConnexion, $requete);
        } else {
            $url = FALSE;
        }
        return $url;
    }?>
```

REMARQUE

Utilisation du bouton retour

Si, après avoir suivi un lien, vous revenez sur la page compteurclic.php en utilisant le bouton retour (back), le contenu de la page ne sera pas réactualisé. Par conséquent, le nombre de visites affiché ne sera pas correct. N'oubliez donc pas, dans ce cas, de rafraîchir (bouton actualiser) la page.

SuperTheque

L'essentiel du code de l'application a été présenté en introduction de ce chapitre. Il ne restait plus qu'a connaître les fonctions SQLite pour implémenter l'interface `RessourceInterface` c'est désormais chose faite:

Listing 10.86 : RessourceSQLite_class.php

```
<?php
include_once("RessourceInterface_class.php");
include_once(dirname(__FILE__)."/../config/sqlite_cfg.php");
/**
 * RessourceSQLite_class.php
 * Classe d'acces a une base de donnees SQLite
```

```
 * Cette classe doit implementer toutes les methodes de l'interface
 *   RessourceInterface.
 * Compatibilite: PHP 5
 */
class Ressource implements RessourceInterface
{
    var $idConnexion;

    public function connexion()
    {
        global $sqlite_base;

        if (!isset($this->idConnexion)) {
            $idConnexion = sqlite_popen($sqlite_base);
            if (!$idConnexion) return FALSE;
            $this->idConnexion = $idConnexion;
        }
        return TRUE;
    }

    public function deconnexion()
    {
        // Rien a faire, il s'agit d'une connexion persistante
    }

    public function reset()
    {
        $requete = "DROP TABLE types";
        $idResultat = @sqlite_query($requete, $this->idConnexion);
        // Pas de raison de retourner un erreur
        // si la suppression echoue (si la table n'existe pas)

        $requete = "CREATE TABLE types (".
                        "id INTEGER PRIMARY KEY,".
                        "type VARCHAR(255))";
        $idResultat = sqlite_query($requete, $this->idConnexion);
        if (!$idResultat) return FALSE;

        $types = array("Album", "Film", "Livre", "Musique");
        foreach ($types as $type) {
            $requete = "INSERT INTO types (type) VALUES ('$type')";
            $idResultat = sqlite_query($requete, $this->idConnexion);
            if (!$idResultat) return FALSE;
        }

        $requete = "DROP TABLE articles";
        $idResultat = @sqlite_query($requete, $this->idConnexion);
        // Pas de raison de retourner un erreur
        // si la suppression echoue (si la table n'existe pas)

        $requete = "CREATE TABLE articles (".
                     "id INTEGER PRIMARY KEY,".
```

```
                "albumId INTEGER,".
                "titre VARCHAR(255),".
                "typeId INTEGER,".
                "commentaire VARCHAR(255))";
    $idResultat = sqlite_query($requete, $this->idConnexion);
    if (!$idResultat) return FALSE;
}

public function addArticle($albumId,
                           $titre,
                           $typeId,
                           $commentaire)
{
    $requeteDebut = "INSERT INTO articles (albumId, titre, typeId";
    $requeteFin = ") VALUES ($albumId,".
                            "'".sqlite_escape_string($titre)."',".
                            "$typeId";
    if ($commentaire != "") {
         $requeteDebut .= ",commentaire";
         $requeteFin .= ",'".sqlite_escape_string($commentaire)."'";
    }
    $requete = $requeteDebut . $requeteFin . ")";
    $idResultat = sqlite_query($requete, $this->idConnexion);
    if (!$idResultat) return FALSE;
}

public function getArticle($articleId)
{
    $requete = "SELECT * FROM articles WHERE id=$articleId";
    $idResultat = sqlite_query($requete, $this->idConnexion);
    if (!$idResultat) return FALSE;

    $article = NULL;
    if ($enreg = sqlite_fetch_array($idResultat)) {
        $article = $this->enreg2Article($enreg);
    }
    return $article;
}

public function getArticles(Filtre $filtre, Plage $plage, Tri $tri)
{
    $requete = "SELECT * FROM articles";
    if ($filtre->getAlbumId() != -1) {
        $conditionSQL = "albumId=".$filtre->getAlbumId();
    }
    if ($filtre->getTitre() != "") {
        $conditionSQL = $conditionSQL.
                        " AND titre LIKE '".
                        sqlite_escape_string($filtre->getTitre())."'";
    }
    if ($filtre->getTypeId() != -1) {
        $conditionSQL = $conditionSQL.
```

```
                    " AND typeId=".$filtre->getTypeId();
    }

    if ($tri->getSens() == -1) {
        $triSQL = "ORDER BY ".$tri->getChamp()." DESC";
    } else if ($tri->getSens() == 1) {
        $triSQL = "ORDER BY ".$tri->getChamp()." ASC";
    }

    $limiteSQL = "LIMIT ".$plage->getNbArticleMax().
                 " OFFSET ".$plage->getPremierArticle();

    $requete = $requete." WHERE ".$conditionSQL." ".
                                  $triSQL." ".$limiteSQL;
    $idResultat = sqlite_query($requete, $this->idConnexion);
    if (!$idResultat) return FALSE;

    $articles = NULL;
    while ($enreg = sqlite_fetch_array($idResultat)) {
        $articles[] = $this->enreg2Article($enreg);
    }
    return $articles;
}

public function getNbTotalArticles(Filtre $filtre)
{
    $requete = "SELECT COUNT(*) FROM articles";
    if ($filtre->getAlbumId() != -1) {
        $conditionSQL = "albumId=".$filtre->getAlbumId();
    }
    if ($filtre->getTitre() != "") {
        $conditionSQL = $conditionSQL.
                    " AND titre LIKE '".
                    sqlite_escape_string($filtre->getTitre())."'";
    }
    if ($filtre->getTypeId() != -1) {
        $conditionSQL = $conditionSQL.
                    " AND typeId=".$filtre->getTypeId();
    }

    $requete = $requete." WHERE ".$conditionSQL;
    $idResultat = sqlite_query($requete, $this->idConnexion);
    if (!$idResultat) return FALSE;

    if ($enreg = sqlite_fetch_array($idResultat)) {
        return $enreg[0];
    } else return FALSE;
}

public function getTypes()
{
    $requete = "SELECT * FROM types";
```

```
        $idResultat = sqlite_query($requete, $this->idConnexion);
        if (!$idResultat) return FALSE;

        $types = NULL;
        while ($enreg = sqlite_fetch_array($idResultat)) {
            $types[$enreg["id"]] = $enreg["type"];
        }
        return $types;
    }

    public function getAlbumTypeId()
    {
        // Avec SQLite les champs auto-incrementes commencent a 1
        return 1;
    }


    private function enreg2Article($enreg)
    {
        $article = new Article();
        $article->setId($enreg["id"]);
        $article->setAlbumId($enreg["albumId"]);
        $article->setTitre($enreg["titre"]);
        $article->setTypeId($enreg["typeId"]);
        $article->setCommentaire($enreg["commentaire"]);
        return $article;
    }
}
?>
```

Comme vous le constatez, la principale difficulté n'est pas tant au niveau des fonctions disponibles qu'au niveau de la construction des requêtes SQL. La requête sera simple lorsqu'il ne s'agit que de récupérer un enregistrement identifié par la clé `Id` (comme c'est le cas avec `getArticle()`) elle sera bien plus complexe s'il s'agit des récupérer un ensemble d'enregistrements répondant à des critères précis et triés (comme c'est le cas avec `getArticles()`).

Requêtes avec des apostrophes

Lors de la construction de la requête, il faut bien garder à l'esprit que l'un des éléments de la requête (typiquement un champ de type texte saisi par l'utilisateur, comme ici le titre ou le commentaire) peut contenir une apostrophe qui pourrait être confondu avec le délimiteur de chaîne. Pensez donc bien à faire un appel à `sqlite_escape_string()` lorsque cela peut s'avérer nécessaire.

Affichage page par page

La meilleure façon avec SQLite pour n'extraire qu'un sous-ensemble des enregistrements retournés par une requête SQL consiste à utiliser les instructions `LIMIT` et `OFFSET`. `LIMIT` permet de préciser combien d'enregistrements doivent être retournés et `OFFSET` à partir duquel commencer (sachant que le premier enregistrement porte l'index 0).

Ainsi, `SELECT * FROM nomtable LIMIT 10 OFFSET 0;` retourne les dix premiers enregistrements (de l'index 0 à l'index 9) et `SELECT * FROM nomtable LIMIT 20 OFFSET 10;` retourne les vingt suivants (de l'index 10 à l'index 29).

Tri

Modifier l'ordre d'affichage des annonces n'est pas non plus bien sorcier. Le tri peut s'effectuer directement au niveau de la requête SQL, grâce à l'instruction `ORDER BY`.

Moteur de recherche (filtre)

Pour filtrer, il suffit d'utiliser l'instruction `WHERE`. Dans notre cas, nous souhaitons autoriser l'utilisation de jokers (comme %) afin, par exemple, de rechercher les titres commençant par une chaîne donnée. C'est pourquoi, nous ne ferons pas un test de type `titre='motcle'` mais `titre LIKE 'motcle'` (où `motcle` pourrait avoir la valeur "La 7eme compagnie%").

En savoir plus...

Dans le cas d'une requête mise en mémoire (c'est à dire `sqlite_query()` ou `sqlite_array_query()`), il est possible de connaître le nombre de ligne du résultat.

sqlite_num_rows()

Permet de compter le nombre de lignes du résultat.

Syntaxe	`int sqlite_num_rows(resource $idRequete)`
`$idRequete`	Identifiant de requête.
`retour`	Nombre de lignes du résultat.

Il est également possible de connaître le nombre de colonnes en faisant appel à `sqlite_num_fields()`.

sqlite_num_fields()

Permet de compter le nombre de colonnes du résultat.

Syntaxe	`int sqlite_num_fields(resource $idRequete)`
`$idRequete`	Identifiant de requête.
`retour`	Nombre de colonnes du résultat.

Pour connaître le nom d'une colonne par son indice, il suffit de faire appel à `sqlite_field_name()`.

sqlite_field_name()

Retourne le nom d'une colonne

Syntaxe	`int sqlite_field_name(resource $idRequete, int numeroColonne)`
`$idRequete`	Identifiant de requête.
`$numeroColonne`	Numéro de la colonne dont on veut le nom.
`retour`	Nom de la colonne.

Autre fonction très utile, `sqlite_last_insert_rowid()` permet de connaître l'identifiant qui a été automatiquement attribué à la dernière insertion avec une colonne auto-incrémentée.

sqlite_last_insert_rowid()

Permet de récupérer l'identifiant automatiquement attribué.

Syntaxe	`int sqlite_last_insert_rowid(resource $idConnexion)`
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `sqlite_open()` ou `sqlite_popen()`.
`retour`	Identifiant attribué.

Lors d'une mise à jour il peut être utile de savoir combien de lignes ont été modifiées en utilisant `sqlite_changes()`.

sqlite_changes()

Retourne le nombre de lignes modifiées par la dernière requête effectuée sur la base de données dont l'identifiant est passé en paramètre.

Syntaxe	`int sqlite_changes(resource $idConnexion)`
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `sqlite_open()` ou `sqlite_popen()`.
`retour`	Nombre de lignes modifiées.

Lorsque un accès à la base de données doit-être effectué, le serveur attend que le fichier soit libre (non utilisé par un autre processus). Par défaut au bout de soixante secondes, le serveur génère une erreur. Il est possible de changer cette valeur en utilisant `sqlite_busy_timeout()`.

sqlite_busy_timeout()

Permet de définir le temps avant lequel le serveur de base de données génère une erreur s'il n'arrive pas à accéder au fichier.

Syntaxe	`void sqlite_busy_timeout(resource $idConnexion, int $temps)`
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `sqlite_open()` ou `sqlite_popen()`.
`$temps`	Temps en millisecond avant génération d'une erreur.

Gérer l'encodage

Pour encoder des données binaires (qui pourrait inclure le caractère NUL au milieu des données) il faut utiliser `sqlite_udf_encode_binary()`.

sqlite_udf_encode_binary()

Encode la donnée.

Syntaxe	`string sqlite_udf_encode_binary(string $donnee)`
`$donnee`	Donnée à encoder.
`retour`	Donnée encodée.

Lors de la récupération de cette donnée, il faut faire l'opération inverse en utilisant `sqlite_udf_decode_binary()`.

sqlite_udf_decode_binary()

Décode la donnée.

Syntaxe	`string sqlite_udf_decode_binary(string $donnee)`
`$donnee`	Donnée à décoder.
`retour`	Donnée décodée.

`sqlite_escape_string()` permet en outre d'éviter les problèmes liés aux apostrophes.

sqlite_escape_string()

Double les apostrophes et rend les données binaires

Syntaxe	`string sqlite_escape_string(string $donnee)`
`$donnee`	Donnée à encoder.
`retour`	Donnée encodée.

A propos de SQLite

Il est possible de récupérer des informations sur la bibliothèque installée.

sqlite_libversion()

Retourne la version de SQLite utilisée.

Syntaxe	`string sqlite_libversion()`
`retour`	Numéro de version.

sqlite_libencoding()

Retourne l'encodage utilisé. (ISO-8859-1 ou UTF-8)

Syntaxe	`string sqlite_libencoding()`
`retour`	Encodage utilisé (iso8859 ou utf8).

10.13. SQL Server (MS)

SQL Server est la solution de base de données proposée par Microsoft pour ses systèmes d'exploitation Windows NT et XP.

Installation

Microsoft SQL Server n'existe, bien entendu, que sous Windows.

Nous nous contenterons ici de décrire une installation standard, sans prendre en compte les problèmes d'optimisation et de sécurité. Le but étant que vous puissiez installer PHP et SQL Server sur des machines de test pour vous familiariser avec cet environnement avant de passer à un serveur de bases de données destiné à la production.

Pré-requis

Pour commencer, vous devez vous procurer le CD-ROM d'installation ou télécharger une version d'évaluation à l'adresse **http://www.microsoft.com/sql/downloads/default.asp**.

Installation du serveur de bases de données

Nous supposerons ici que le serveur de bases de données est installé sur la même machine que le serveur web. La version testée est la version Microsoft SQL Server 2000 de démonstration.

La première des choses à faire est de lancer le fichier *autorun.exe* de votre dossier.

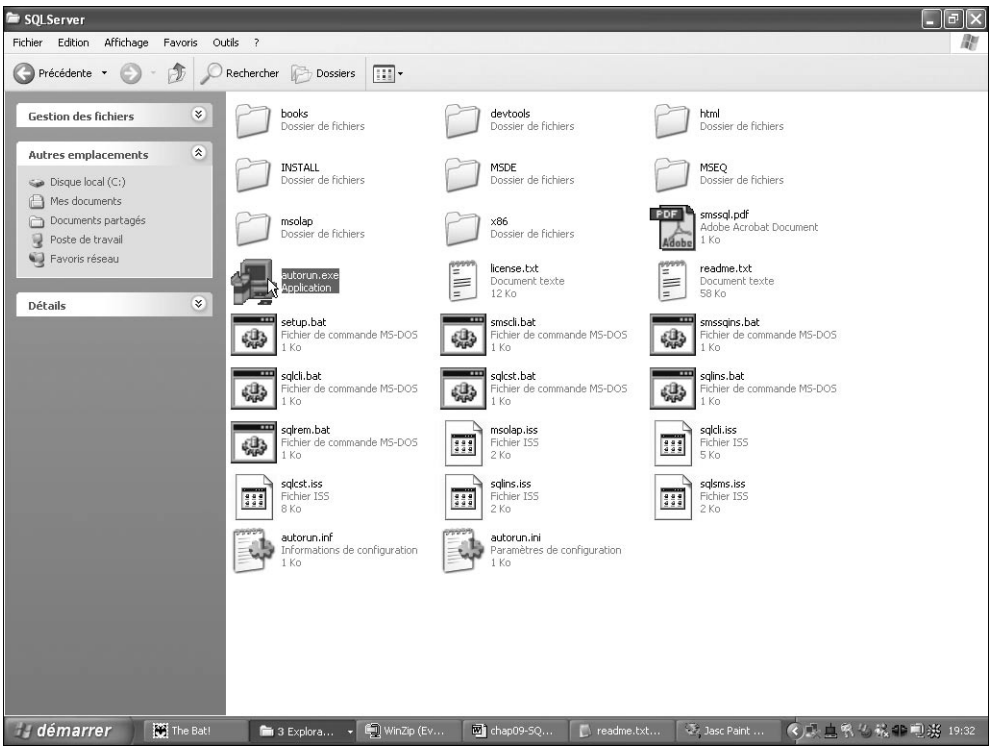


Figure 10.39 : *Dossier d'installation*

Il faut ensuite cliquer sur **Composants** de SQL Server 2000, puis *Installer le serveur*.

Le programme d'installation demandera alors si le serveur est à installer sur la machine courante ou sur une machine distante. Ici, l'installation se fera sur la machine courante.

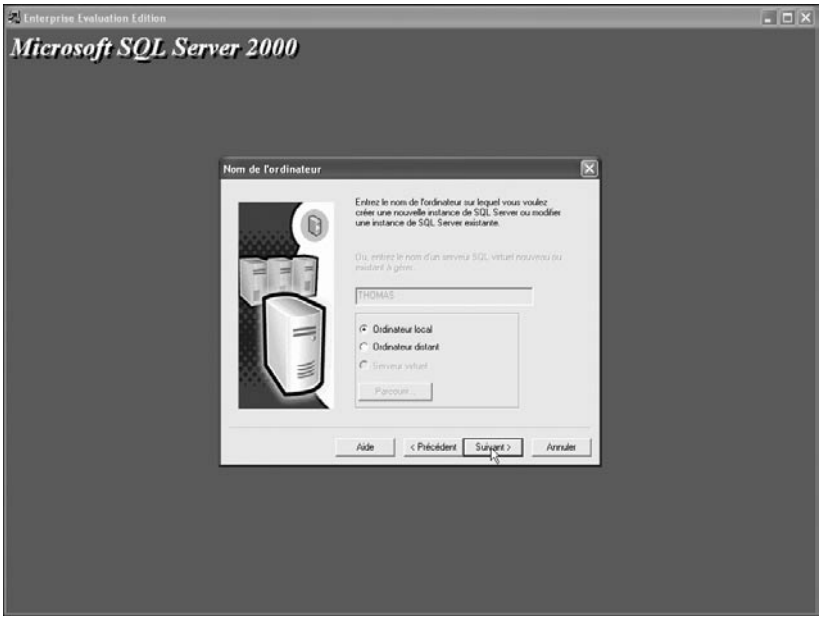


Figure 10.40 : *Emplacement du serveur*

Il faudra ensuite choisir entre la configuration par défaut et la configuration personnalisée. Nous prendrons celle par défaut. Le nom et la société de l'utilisateur sont ensuite demandés.

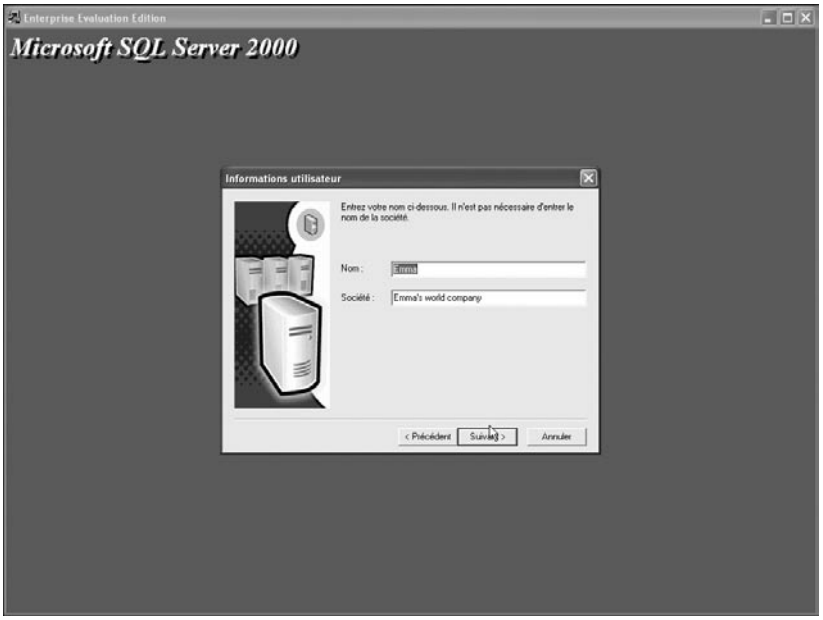


Figure 10.41 : *Nom et société*

Nous choisirons ensuite d'installer à la fois le serveur et les outils clients.

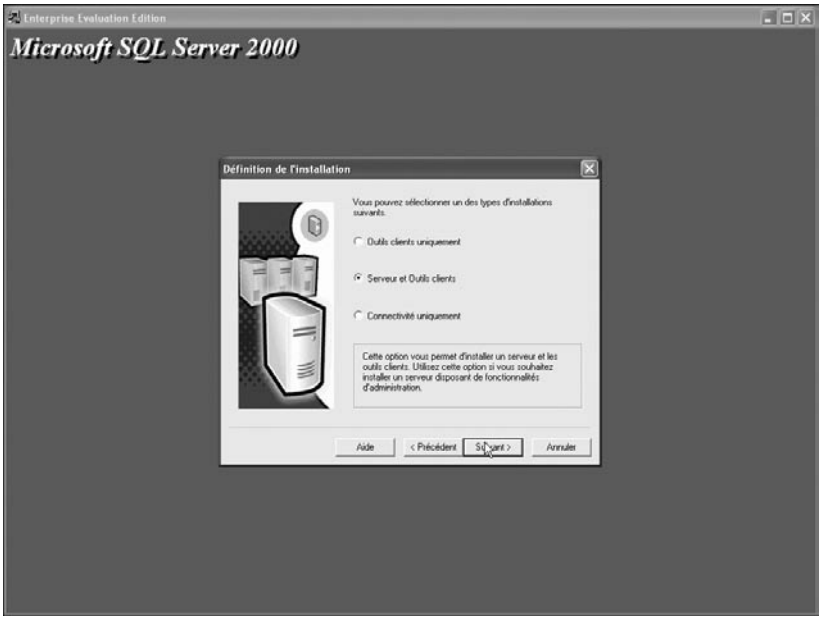


Figure 10.42 : *Type d'installation ½*

Les options d'installation sont ensuite configurables. Nous garderons les valeurs par défaut :

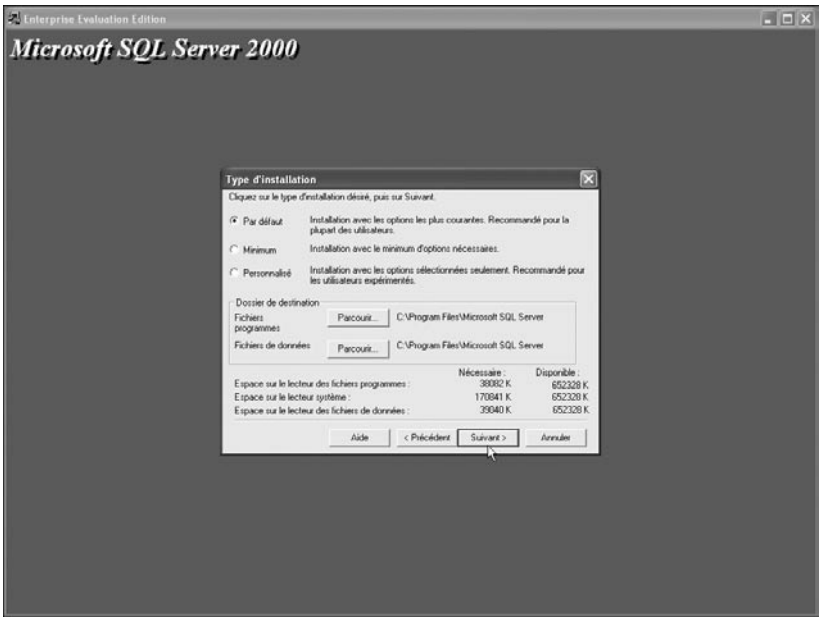


Figure 10.43 : *Type d'installation 2/2*

L'administrateur du service est à définir : par défaut, cela peut être l'utilisateur qui installe le serveur, ou bien un autre qui peut être sur un domaine différent.

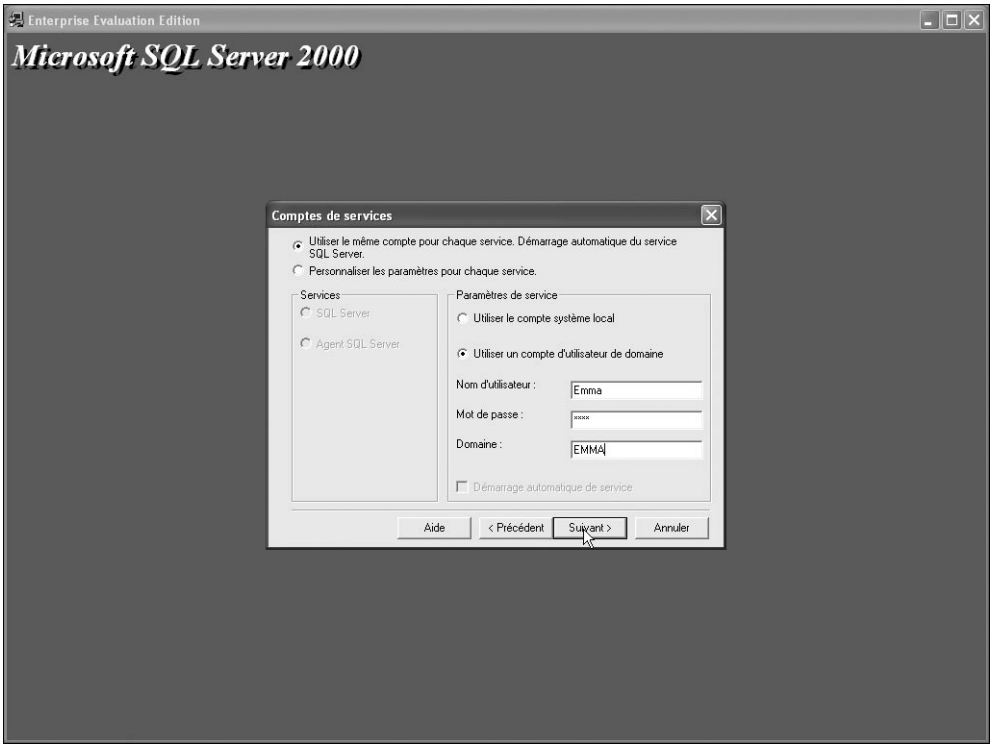


Figure 10.44 : *Administrateur*

Pour finir, il est demandé de préciser le mode d'authentification. Il faut absolument mettre les deux modes d'authentification (SQL serveur et Windows), sinon la connexion par PHP ne fonctionnera pas.

Démarrage du serveur de bases de données

Le serveur s'étant installé en tant que service, son lancement est automatique.

Création d'une base

Pour créer une base de données, il faut tout d'abord lancer le programme appelé *SQL Server Enterprise Manager*, puis ouvrir les dossiers jusqu'à *Bases de données*. Là, un clic bouton droit sur ce dossier fait apparaître un menu contextuel affichant *Nouvelle base de données*.

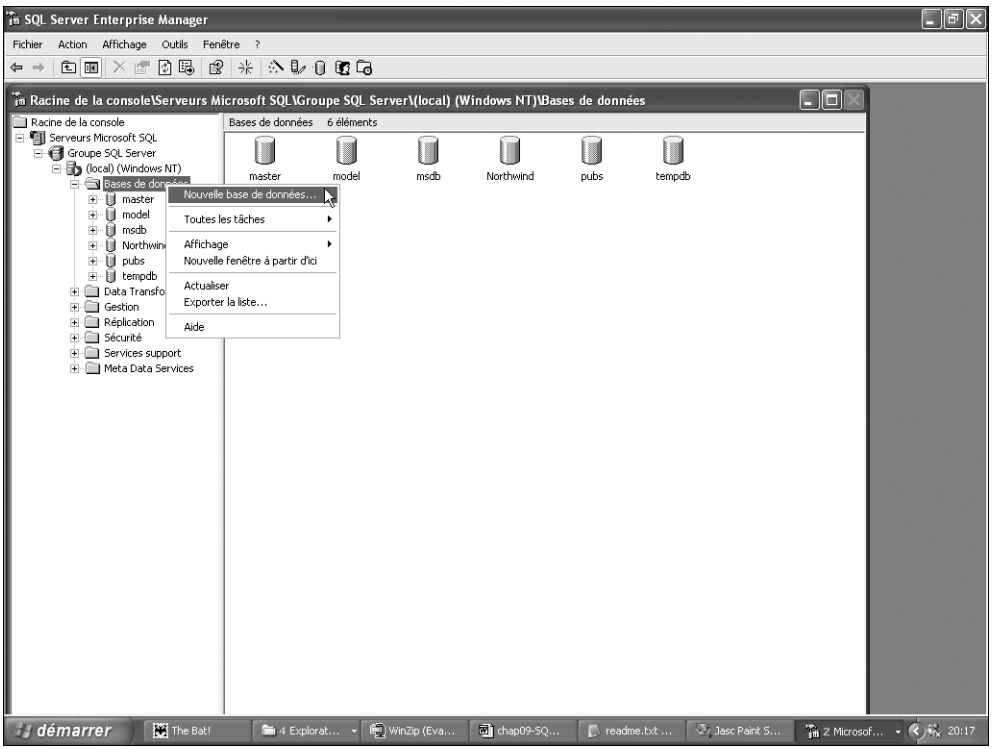


Figure 10.45 : *Ajouter une base de données*

Il suffit alors de donner un nom à la base de données et, éventuellement, de modifier les chemins pour sauvegarder les données et les fichiers de traces.

Ajouter un utilisateur

Pour ajouter un utilisateur, il faut d'abord créer une connexion. Vous devez alors saisir un nom, sélectionner *Authentification SQL Server*, entrer un mot de passe, puis sélectionner la base de données qui vous intéresse (voir fig. 10.46).

Ensuite, pour ajouter un utilisateur à la base de données, il faut aller sur l'onglet **Accès aux bases de données** et cliquer sur les bases auxquelles vous souhaitez donner accès, puis choisir les droits d'accès.

Dans le cas présent, `public`, `db_owner`, `db_datareader` et `db_datawriter` ont été sélectionnés.

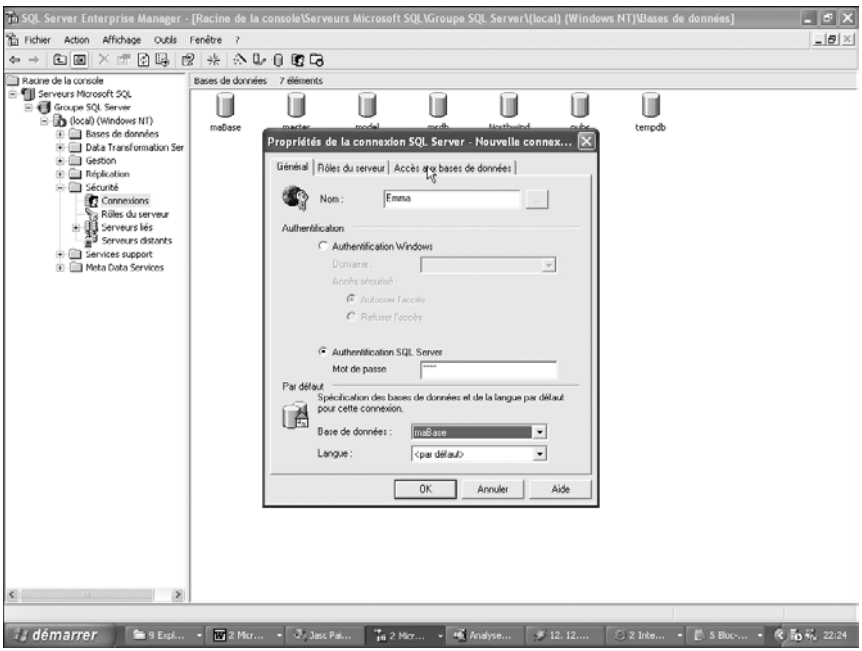


Figure 10.46 : *Ajout d'une connexion*

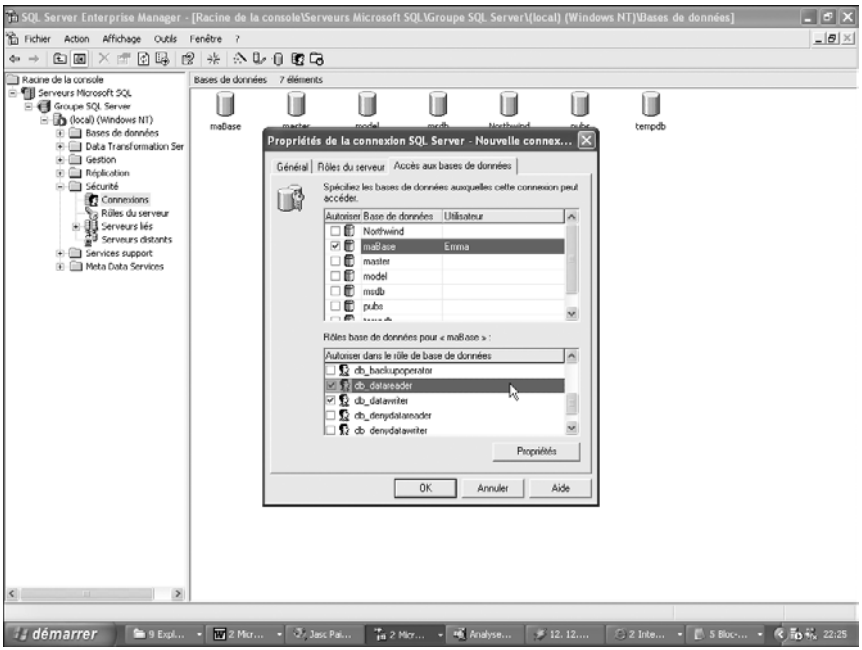


Figure 10.47 : *Ajout d'un utilisateur*

Test du serveur de bases de données

Pour tester simplement que le serveur est bien installé, il suffit de lancer l'*Analyseur de requêtes*.

Après avoir sélectionné la bonne base dans le menu déroulant, il suffit d'écrire une ou plusieurs requêtes puis de lancer l'exécution par F5 dans la fenêtre **Requête**. Normalement, un message apparaîtra dans la fenêtre de la requête.

Pour vérifier, il suffit d'aller ouvrir le répertoire de la base de données, puis le répertoire *Tables utilisateur*. Pour ouvrir la table précédemment créée, il suffit de cliquer avec le bouton droit dessus et de faire *Ouvrir*. Le ou les enregistrements apparaîtront dans une nouvelle fenêtre.

Configuration de PHP avec support SQLServer

L'extension MS SQL n'est disponible que sur les systèmes Windows 32 bits. Pour les autres plateformes, il est tout de même possible d'interroger une base de données MS SQL à l'aide de l'extension Sybase.

Vous devez, dans un premier temps, vous assurer d'avoir un fichier *php_mssql.dll* (fourni aussi bien avec l'archive du PHP Group qu'avec EasyPHP) dans le répertoire de vos extensions PHP. Il vous suffit ensuite de modifier le fichier *php.ini* pour ajouter ou décommenter une ligne.

```
extension=php_mssql.dll
```

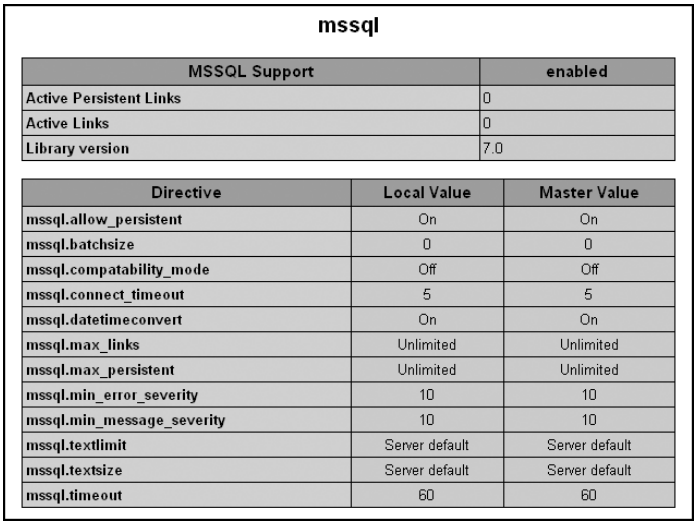
Quelques paramètres sont configurables dans le fichier *php.ini* :

```
[MSSQL]
; Permet ou interdit les connexions persistantes.
mssql.allow_persistent = On

; Nombre maximum de connexions persistantes (-1 pour illimité).
mssql.max_persistent = -1

; Nombre maximum de connexions persistantes et non persistantes
; (-1 pour illimité).
mssql.max_links = -1

; Sévérité minimum des erreurs à afficher.
mssql.min_error_severity = 10

; Sévérité minimum des messages à afficher.
mssql.min_message_severity = 10

; Compataibilité avec la version 3.0 de PHP
mssql.compatability_mode = Off

; Plage valide  0 - 2147483647.  Par défaut = 4096.
;mssql.textlimit = 4096

; Plage valide  0 - 2147483647.  Par défaut = 4096.
;mssql.textsize = 4096

; Limit le nombre d'enregistrement par groupe.
```

```
; 0 = tous les enregistrements dans le même groupe.
;mssql.batchsize = 0
```

Vérification

Vous pouvez maintenant tester un script ne contenant que le code `<?php phpinfo(); ?>`. Vous devez apercevoir les lignes suivantes :



mssql

MSSQL Support	enabled
Active Persistent Links	0
Active Links	0
Library version	7.0

Directive	Local Value	Master Value
mssql.allow_persistent	On	On
mssql.batchsize	0	0
mssql.compatability_mode	Off	Off
mssql.connect_timeout	5	5
mssql.datetimeconvert	On	On
mssql.max_links	Unlimited	Unlimited
mssql.max_persistent	Unlimited	Unlimited
mssql.min_error_severity	10	10
mssql.min_message_severity	10	10
mssql.textlimit	Server default	Server default
mssql.textsize	Server default	Server default
mssql.timeout	60	60

Figure 10.48 : *phpinfo()*

Utilisation

L'utilisation basique de la base de données s'opère selon le schéma suivant :

- Connexion grâce aux fonctions `mssql_connect()` ou `mssql_pconnect()` et `mssql_select_db()`.
- Soumission de la requête via la fonction `mssql_query()`.
- Déconnexion grâce à la fonction `mssql_close()` (dans le cas d'une connexion non persistante).

Connexion

La connexion à une base de données Microsoft SQL Server nécessite deux opérations. Il faut, dans un premier temps, se connecter au serveur de bases de données avec la fonction `mssql_connect()`; la sélection de la base s'opère dans un second temps avec `mssql_select_db()`.

mssql_connect()

Établit une connexion (non persistante) avec le serveur de bases de données.

Syntaxe	`resource mssql_connect([string $nomServeur [:$port] [, string $utilisateur [, $motDePasse]]])`
`$nomServeur`	Nom du serveur (par défaut "`localhost`").
`$port`	Port sur lequel la connexion au serveur de bases de données s'effectue.
`$utilisateur`	Nom d'utilisateur.
`$motDePasse`	Mot de passe de l'utilisateur.
`retour`	Identifiant de connexion au serveur de bases de données, ou `FALSE` en cas d'échec.

L'utilisation d'une connexion persistante requiert, à la place, l'appel à la fonction `mssql_pconnect()`.

mssql_pconnect()

Établit une connexion persistante avec le serveur de bases de données. Si une connexion persistante est disponible, elle sera utilisée ; sinon, une nouvelle connexion persistante sera créée.

Syntaxe	`resource mssql_pconnect([string $nomServeur [:$port] [, string $utilisateur [, $motDePasse]]])`
`$nomServeur`	Nom du serveur (par défaut "`localhost`").
`$port`	Port sur lequel la connexion au serveur de bases de données s'effectue.
`$utilisateur`	Nom d'utilisateur.
`$motDePasse`	Mot de passe de l'utilisateur.
`retour`	Identifiant de connexion au serveur de bases de données, ou `FALSE` en cas d'échec.

mssql_select_db()

Sélectionne une base de données.

Syntaxe	`boolean mssql_select_db(string $baseDonnees [, resource $idConnexion])`
`$baseDonnees`	Nom de la base de données.

`$idConnexion`	Identifiant de connexion au serveur de bases de données tel que retourné par `mssql_connect()`. Par défaut, c'est l'identifiant de la dernière connexion qui est utilisé.
`retour`	`TRUE` en cas de succès, `FALSE` sinon (ex. : base de données inexistante).

Exécution de la requête SQL

L'exécution d'une requête SQL s'opère par un simple appel à `mssql_query()`.

mssql_query()

Exécute une requête SQL.

Syntaxe	`resource mssql_query(string $requete [, resource $idConnexion])`
`$requete`	Requête SQL.
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `mssql_connect()` ou `mssql_pconnect()`. Par défaut, c'est l'identifiant de la dernière connexion qui est utilisé.
`retour`	Identifiant de requête SQL, ou `FALSE` en cas d'échec.

Dans le cas d'une requête retournant un résultat (typiquement une requête `SELECT`), il faudra également analyser le résultat. Mais nous verrons cela un peu plus loin.

Déconnexion

Pour se déconnecter – opération théoriquement facultative mais toutefois vivement conseillée –, vous disposez de la fonction `mssql_close()`.

mssql_close()

Met fin à la connexion à la base de données et libère les ressources associées.

Syntaxe	`boolean mssql_close([resource $idConnexion])`
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `mssql_connect()`. Par défaut, c'est l'identifiant de la dernière connexion qui est utilisé.
`retour`	`TRUE` si l'opération a été effectuée avec succès, `FALSE` en cas d'échec.

Premier exemple

Nous voilà donc à même de construire notre premier script utilisant une base de données.


ATTENTION

Vérifiez les paramètres
Prenez garde ! Avant d'exécuter ce script, assurez-vous que les paramètres prédéfinis ne risquent pas de conduire à la suppression des données d'une de vos bases qui, par coïncidence, existerait déjà.

Comme cela est fortement conseillé, nous avons, ici, isolé les paramètres de connexion dans un fichier aisément repérable.

Listing 10.87 : parametres_bd_inc.php

```
<?php

    // Paramètres de connexion à la base de données
    // N'hésitez pas à les modifier selon vos besoins

    $serveur     = "localhost";
    $base        = "maBase";
    $utilisateur = "Emma";
    $motDePasse  = "emma";
?>
```

Notre script principal sera donc :

Listing 10.88 : insert01.php

```
<?php

    // Paramètres du script
    require_once("parametres_bd_inc.php");
    $table = "maTable";

    // Inclusion du script contenant les fonctions
    require_once("insert01_bd_inc.php");

    // Connexion à la base de données
    $idConnexion = mssql_pconnect($serveur, $utilisateur, $motDePasse);
    if (!$idConnexion) {
        die("<b>Impossible de se connecter à la base de données</b>");
    }
    if (!mssql_select_db($base)) {
        die("<b>Impossible de se connecter à la base de données</b>");
    }

    // Appel de la fonction principale
    if (EX_initialiseBD($idConnexion, $table)) {
        echo "Voilà, une nouvelle table avec quelques données, ".
```

```
            "vous pouvez le vérifier avec votre client de base ".
            " de données ou via les scripts suivants.";
    } else {
        echo "La création ou l'alimentation de la table à échouée.";
    }

    // Pas de déconnexion dans le cas d'une connexion persistante
    // mssql_close($idConnexion);
?>
```

et nécessitera :

Listing 10.89 : insert01_bd_inc.php

```
<?php

    /**
     * Fonction chargée de créer et d'alimenter une table
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la table
     **/
    function EX_initialiseBD($idConnexion, $table)
    {
        // Supprime la précédente table
        $requete = "DROP TABLE $table";
        @mssql_query($requete, $idConnexion);

        // Crée la table
        $requete = "CREATE TABLE $table (filmId int ".
                                            " IDENTITY PRIMARY KEY,".
                                        "film VARCHAR(64))";
        if (!mssql_query($requete, $idConnexion)) return FALSE;

        // Ajoute quelques données
        $requete = "INSERT INTO $table (film) VALUES ('Forrest Gump')";
        if (!mssql_query($requete, $idConnexion)) return FALSE;
        $requete = "INSERT INTO $table (film) VALUES ('Matrix')";
        if (!mssql_query($requete, $idConnexion)) return FALSE;
        $requete = "INSERT INTO $table (film) VALUES ('La cité de la peur')";
        if (!mssql_query($requete, $idConnexion)) return FALSE;

        return TRUE;
    }
?>
```

Les procédures stockées

Les procédures stockées offrent de multiples avantages, puisqu'elles permettent d'exécuter plusieurs requêtes en un seul appel (une seule interrogation du serveur de bases de données par

le serveur web). De plus, ces requêtes ne sont analysées qu'une seule fois, et non à chaque sollicitation, tout en permettant des appels successifs avec des paramètres différents.

De ce fait, les procédures stockées permettent d'effectuer des requêtes plus rapidement. L'utilisation se justifie particulièrement lorsque les mêmes requêtes sont exécutées de nombreuses fois en ne changeant que quelques valeurs.

Il est possible d'écrire une requête pour produire une procédure stockée ; il est aussi possible d'utiliser un outil d'aide fourni avec Microsoft SQL Server, plus précisément dans le logiciel Enterprise Manager fourni avec SQL Server.

Après avoir ouvert SQL Server Enterprise Manager, il faut choisir **Outils>Assistants**, ouvrir l'onglet **Base de données** , cliquer sur *Assistant création de procédure stockée* puis OK.

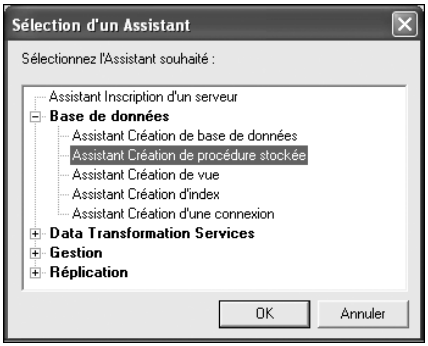


Figure 10.49 : *Assistant*

Il faut ensuite définir sur quelle base de données la procédure stockée doit s'appliquer.

Figure 10.50 : *Base de données impliquée*

Puis il faut définir la ou les tables concernée(s) ainsi que la manière dont elles le sont (Insertion, suppression et/ou mise à jour). Dans notre exemple, nous voulons simplement créer une procédure pour insérer une URL dans la table *compteurclic*.

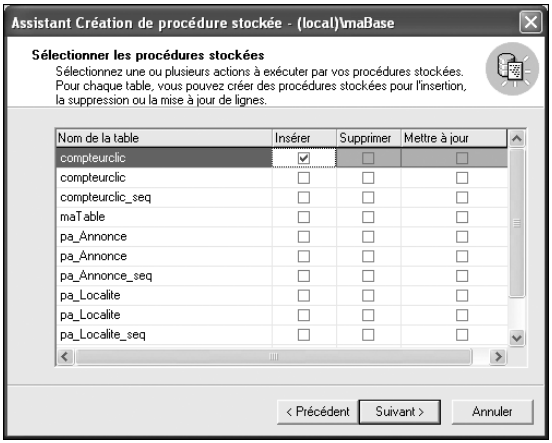


Figure 10.51 : *Tables impliquées*

Il est ensuite possible de modifier les propriétés d'une procédure stockée pour ne sélectionner que les champs que l'on veut passer en paramètre. (Ici, les deux autres champs peuvent être remplis par défaut par la base de données.)

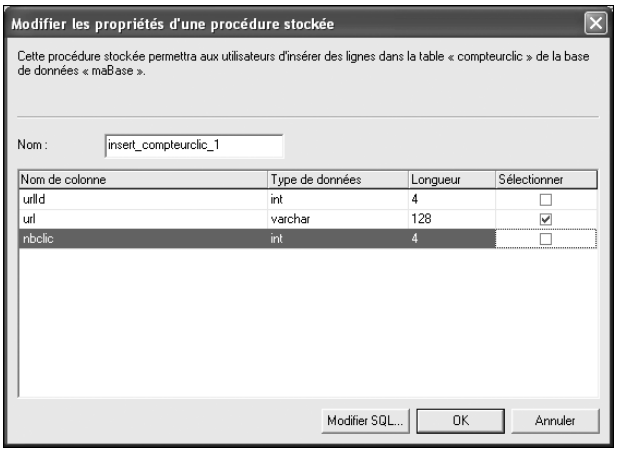


Figure 10.52 : *Détails*

Il est également possible d'intervenir sur la requête SQL. Dans notre cas, puisqu'il ne s'agit que d'une requête d'insertion, cela s'apparente plus à une requête préparée qu'à une véritable procédure stockée. Mais qu'importe ! à titre d'exemple, cela est largement suffisant.

Voici un script qui ajoutera l'URL **http://www.toutestfacile.com** à la table *compteurclic*.

Listing 10.90 : procstock.php

```
<?php
    $idConnexion = mssql_pconnect("localhost", "Emma", "emma");
    if (!$idConnexion) {
        die("<b>Impossible de se connecter à la base de données</b>");
    }
    if (!mssql_select_db("maBase")) {
```

```
        die("<b>Impossible de se connecter à la base de données</b>");
    }
    $url= "http://www.toutestfacile.com";
    $procedure = mssql_init("insert_compteurclic_1");
    mssql_bind($procedure, "@url_1", $url , SQLVARCHAR, FALSE, FALSE, 128);
    mssql_execute($procedure);
?>
```


mssql_init()

Initialise une procédure stockée (distante ou non).

Syntaxe	`int mssql_init(string $nomProcedure [, resource $idConnexion]);`
`$nomProcedure`	Nom de la procédure à exécuter.
`$idConnexion`	Identifiant de connexion à une base de données tel que retourné par `mssql_connect()`. Par défaut, c'est l'identifiant de la dernière connexion qui est utilisé.
`retour`	Un identifiant de procédure stockée.

mssql_bind()

Ajoute un paramètre à une procédure stockée (distante ou non).

Syntaxe	`boolean mssql_bind(int $idProcedure, int $nomParametre, mixed $variable, int $type [, boolean $paramDeSortie [, boolean $null [, int $longMax]]])`
`$idProcedure`	Identifiant de procédure stockée tel que retourné par `mssql_init()`.
`$nomParametre`	Le nom du paramètre précédé du signe '@' définissant une variable dans les procédures stockées de MS SQL.
`$variable`	Elle peut être passée par valeur ou par référence.
`$type`	Le type du paramètre, au choix parmi : `SQLTEXT`, `SQLVARCHAR`, `SQLCHAR`, `SQLINT1`, `SQLINT2`, `SQLINT4`, `SQLBIT` et `SQLFLT8`.
`$paramDeSortie`	`TRUE` s'il s'agit d'un paramètre de sortie, `FALSE` (par défaut) sinon.
`$null`	`TRUE` si le paramètre vaut `NULL`, `FALSE` (par défaut)sinon.
`$longMax`	Longueur du champ (`-1` par défaut). Ce paramètre est utilisé pour les types `CHAR` et `VARCHAR` ; cette valeur doit être la même que celle définie lors de la création de la table.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

mssql_execute()

Exécute une procédure stockée sur une base de données Microsoft SQL Server.

Syntaxe	`mixed mssql_execute(int $idProcedure)`
`$idProcedure`	Identifiant de procédure stockée tel que retourné par `mssql_init()`.
`retour`	Identifiant de résultat de requête si la procédure retourne un résultat, `TRUE` si la procédure ne retourne pas de résultat, `FALSE` en cas d'erreur.

Lecture des enregistrements

Dans le cas d'une requête retournant une liste de données (typiquement un `SELECT`), il convient de récupérer l'identifiant de requête et de l'utiliser généralement pour lire, en boucle, ligne après ligne, chaque enregistrement (ou l'ensemble d'un bloc). Il existe diverses fonctions, chacune permettant de récupérer les champs des enregistrements sous différentes formes.

Les informations peuvent ainsi être retournées :

- Champ par champ ;
- Enregistrement par enregistrement (sous la forme d'un tableau indexé, associatif, ou à la fois indexé et associatif).

Lecture champ par champ

Bien que cela présente peu d'intérêt, sachez qu'il existe une fonction appelée `mssql_result()` permettant d'accéder au résultat champ par champ.

mssql_result()

Retourne un champ d'un enregistrement.

Syntaxe	`string mssql_result(resource $resultat, int $enregistrement, mixed $champ)`
`$resultat`	Identifiant de résultat tel que retourné par `mssql_query()`.
`$enregistrement`	Numéro de l'enregistrement.
`$champ`	Numéro ou nom du champ.
`retour`	Le champ demandé, ou `FALSE` en cas d'erreur.

Vitesse

`mssql_result()` est une fonction TRÈS lente par rapport à ses homologues `mssql_fetch_row()`, `mssql_fetch_array()` et `mssql_fetch_object()`.

Lecture enregistrement par enregistrement

La forme la plus simple est celle retournée par la fonction `mssql_fetch_row()`.

mssql_fetch_row()

Retourne un enregistrement, retourné par une requête SQL, sous la forme d'un tableau indexé.

Syntaxe	`array mssql_fetch_row(resource $idResultat)`
`$idResultat`	Identifiant de résultat tel que retourné par `mssql_query()`.
`retour`	Tableau indexé contenant autant d'éléments que de champs retournés par la requête, ou `FALSE` s'il n'y a plus d'enregistrement à lire.

Listing 10.91 : select_eei_bd_inc.php

```
<?php

    /**
     * Fonction listant le contenu d'une table
     * contenant 2 champs (filmId et film)
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la table
     **/
    function EX_listeContenu($idConnexion, $table)
    {
        // Requete
        $requete = "SELECT * FROM $table";
        $idResultat = mssql_query($requete, $idConnexion);
        if (!$idResultat) return FALSE;

        // Boucle de lecture (et d'affichage) des enregistrements
        while ($enreg = mssql_fetch_row($idResultat)) {
            echo "FilmId=".$enreg[0]." ".
                 "Film =" .$enreg[1]."<br />";
        }
        return TRUE;
     }
?>
```

Mais il est également possible de retourner un enregistrement sous la forme d'un tableau associatif (où les clés portent les noms des champs), grâce à la fonction `mssql_fetch_assoc()` qui a la même syntaxe que `mssql_fetch_row()`.

Il est surtout possible d'avoir les deux fonctions pour le prix d'une avec la fonction `mssql_fetch_array()`.

mssql_fetch_array()

Retourne un enregistrement, retourné par une requête SQL, sous la forme d'un tableau indexé et/ou associatif.

Syntaxe	`array mssql_fetch_array(resource $idResultat)`
`$idResultat`	Identifiant de résultat tel que retourné par `mssql_query()`.
`retour`	Tableau indexé et/ou associatif, ou `FALSE` s'il n'y a plus d'enregistrement.

C'est ce mode de lecture des données que nous privilégierons. Il est en effet fort pratique pour les champs dont on connaît le nom (ce qui est le plus souvent le cas), et permet également de faire référence à des champs sans nom, comme par exemple un champ *SUM(nomchamp)*. Notez toutefois qu'il est possible d'affecter un nom (ex: *somme*) à ce genre de champ en indiquant `SUM(nomchamp) AS somme`.

L'exemple précédent gagnera ainsi en lisibilité et deviendra :

Listing 10.92 : select_eea_bd_inc.php

```
<?php

    /**
     * Fonction listant le contenu d'une table
     * contenant 2 champs (filmId et film)
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la table
     **/
    function EX_listeContenu($idConnexion, $table)
    {
        // Requete
        $requete = "SELECT * FROM $table";
        $idResultat = mssql_query($requete, $idConnexion);
        if (!$idResultat) return FALSE;

        // Boucle de lecture (et d'affichage) des enregistrements
        while ($enreg = mssql_fetch_array($idResultat)) {
            echo "FilmId=".$enreg["filmId"]." ".
                 "Film =" .$enreg["film"]."<br />";
        }
        return TRUE;
     }
?>
```

Pour votre culture personnelle, sachez qu'il existe également une fonction (d'un intérêt discutable) qui permet de retourner l'enregistrement sous la forme d'un objet possédant des variables internes portant les mêmes noms (en minuscules) que les champs. Cette fonction s'appelle `mssql_fetch_object()` et possède la même syntaxe que `mssql_fetch_row()`, si ce n'est qu'elle retourne un objet et non un tableau.

La boucle de lecture se transforme alors en :

```
<?php
while ($enreg = mssql_fetch_object($idResultat)) {
    echo "FilmId = ".$enreg->filmid." Film = ".$enreg->film."<br />";
}
?>
```

Requêtes multiples

La fonction `mssql_query()` permet de lancer plusieurs requêtes en un unique appel. Il est ainsi possible de faire `mssql_query("SELECT * FROM table1; SELECT * FROM table2")`. L'identifiant de résultat alors retourné est celui de la première requête, mais vous pouvez passer au résultat de la requête suivante grâce à `mssql_next_result()`.

mssql_next_result()

Déplace le pointeur sur le prochain groupe de résultats lorsqu'il y en a plusieurs, par exemple lorsque deux requêtes sont exécutées à la fois.

Syntaxe	`boolean mssql_next_result(resource $resultat)`
`$resultat`	Identifiant de résultat tel que retourné par `mssql_query()`.
`retour`	`TRUE` si un autre groupe de résultat est disponible, `FALSE` sinon.

Danger

Le fait que `mssql_query()` permette d'exécuter plusieurs requêtes en une seule fois constitue un danger potentiel. Si, par exemple, votre serveur est configuré de façon à ce que `magic_quote` soit désactivée et que `addSlashes()` ne soit pas appelé, alors une requête du genre :

```
SELECT * FROM matable WHERE champ='$param';
```

peut se tranformer en :

```
SELECT * FROM matable WHERE champ='blabla'; DROP TABLE matable;
';
```

si l'utilisateur à qui il a été demandé de saisir la valeur de `$param` dans un champ d'un formulaire a saisi `"blabla'; DROP TABLE matable; '"`.

Libération des ressources occupées par une réponse SQL

Il est possible, voire conseillé, de libérer la mémoire occupée par le résultat d'une requête (notamment lorsqu'il est volumineux et que les requêtes sont nombreuses) à l'aide de `mssql_free_result()`.

mssql_free_result()

Cette fonction permet de libérer la mémoire avant la fin du script (elle est automatiquement libérée à la fin d'un script).

Syntaxe	`boolean mssql_free_result(resource $resultat)`
`$idResultat`	Identifiant de résultat tel que retourné par `mssql_query()`.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

Nombre d'enregistrements

Nombre d'enregistrements retournés

Il existe trois façons de déterminer le nombre d'enregistrements retournés par une requête.

Si vous souhaitez uniquement connaître le nombre d'enregistrements, mais ne souhaitez pas immédiatement lire ces enregistrements, alors il vous suffit de faire une requête `COUNT()` comme suit :

Listing 10.93 : count_bd_inc.php

```
<?php

    /**
     * Fonction affichant le nombre d'enregistrements
     * dans une table
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la tabme
     **/
    function EX_compte($idConnexion, $table)
    {
        // Requête
        $requete = "SELECT COUNT(*) FROM $table";
        $idResultat = mssql_query($requete, $idConnexion);

        // Lecture du résultat
        if ($enreg = mssql_fetch_row($idResultat)) {
            echo "Il y a ".$enreg[0]." enregistrements dans la table.<br />";
            return TRUE;
        } else {
            echo "Etes vous sûr que la table $table existe ?<br />";
            return FALSE;
        }
    }
?>
```

Si vous souhaitez connaître le nombre d'enregistrements, mais que vous souhaitez lire les enregistrements sans pour autant avoir besoin au préalable d'en connaître le nombre, il suffit de les compter au fur et à mesure de leur lecture.

Listing 10.94 : count_select_bd_inc.php

```
<?php

    /**
     * Fonction listant le contenu d'une table
     * contenant 2 champs (filmId et film)
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la table
     **/
    function EX_listeContenu($idConnexion, $table)
    {
        // Requete
        $requete = "SELECT * FROM $table";
        $idResultat = mssql_query($requete, $idConnexion);
        if (!$idResultat) return FALSE;

        // Boucle de lecture (et de comptage) des enregistrements
        $nb = 0;
        while ($row = mssql_fetch_row($idResultat)) {
            // Vous pouvez manipuler $enreg à votre guise
            //echo "FilmId=".$enreg[0]." ".
            //     "Film =" .$enreg[1]."<br />";
            $nb++;
        }
        echo "Il y a $nb enregistrements dans la table.<br />";

        return TRUE;
     }
?>
```

Enfin, ce que vous attendez tous : si vous souhaitez connaître le nombre d'enregistrements avant de les lire, vous pouvez faire appel à la fonction `mssql_num_rows()`.

mssql_num_rows()

Retourne le nombre d'enregistrements retournés par la requête SQL.

Syntaxe	`int mssql_num_rows(resource $idResultat)`
`$idResultat`	Identifiant de résultat tel que retourné par `mssql_query()`.
`retour`	Nombre d'enregistrements.

Listing 10.95 : count_num_rows_bd_inc.php

```
<?php

    /**
     * Fonction listant le contenu d'une table
     * contenant 2 champs (filmId et film)
     *
     * @param $idConnexion resource Identifiant de connexion BD
     * @param $table       string   Nom de la table
     **/
    function EX_listeContenu($idConnexion, $table)
    {
        // Requete
        $requete = "SELECT * FROM $table";
        $idResultat = mssql_query($requete, $idConnexion);
        if (!$idResultat) return FALSE;

        echo "Il y a ".mssql_num_rows($idResultat)." enregistrements ".
             "dans la table<br />";

        return TRUE;
     }
?>
```

Sachant qu'une requête `SELECT *` est plus longue qu'une requête `SELECT COUNT(*)`, privilégiez la première méthode si vous n'avez que faire du contenu des enregistrements.

Nombre d'enregistrements modifiés

Pour déterminer le nombre d'enregistrements modifiés, vous pouvez appeler la fonction `mssql_rows_affected()`. Elle prend en argument l'identifiant de connexion, et retourne le nombre d'éléments affectés par la dernière requête.

Gestion des erreurs

Jusque-là, en cas d'erreur, nous nous sommes contentés d'afficher un message générique. Il est cependant possible de déterminer plus précisément l'origine de l'erreur.

Cela est notamment possible en récupérant le dernier message via la fonction `mssql_get_last_message()`.

mssql_get_last_message()

Retourne le code d'erreur du dernier appel.

Syntaxe	`string mssql_get_last_message(void)`
`retour`	Le dernier message obtenu.

Modifier l'apparition des messages d'erreur

Chaque erreur et message possède un degré d'importance. Il est possible de préciser le degré d'importance minimal des erreurs et messages à afficher à l'aide de deux fonctions :

mssql_min_error_severity()

Modifie le degré d'importance minimal des erreurs à afficher.

Syntaxe	`void mssql_min_error_severity(int $degre)`
`$degre`	Degré d'importance minimal.

mssql_min_message_severity()

Modifie le degré d'importance minimal des messages à afficher.

Syntaxe	`void mssql_min_message_severity(int $degre)`
`$degre`	Degré d'importance minimal.

Exemples d'applications

Compteur de clics

Une application courante d'utilisation des bases de données, et simple à mettre en œuvre, consiste à créer un compteur de clics.

En effet, peut-être souhaitez-vous proposer, sur votre site, un certain nombre de liens vers des sites Internet (annuaire web) ou vers des fichiers (bibliothèques de scripts). Peut-être que certains (ou la totalité) de ces liens sont proposés par un partenaire. Bref, tout cela vous incite à savoir quelles sont les pages les plus fréquemment consultées (à connaître les centres d'intérêt des visiteurs) et combien de visiteurs vous avez redirigés vers votre sponsor.

La méthode la plus classique consiste à stocker, en base de données, l'ensemble des liens ainsi proposés, et à remplacer les traditionnels liens de la forme **http://www.domaine.com** par un lien vers un script chargé d'incrémenter le compteur correspondant au site indiqué, et de rediriger le client vers le site grâce à la fonction `header()`.

Pour notre exemple, nous utiliserons une table appelée *url* contenant trois champs :

- *urlid*, identifiant de l'URL (champ auto-incrémenté) ;
- *url*, l'URL proprement dite ;
- *nbclic*, le nombre de clics.

Le script de redirection (appelé ici *compteurclic_redirection.php*) devra alors accepter un paramètre (`urlid`) et devra, à partir de ce paramètre, déterminer quelle est la valeur du champ *url*, incrémenter *nbclic* pour cette URL, et rediriger vers *url*.

Les appels à ce script auront alors la forme **http://localhost/compteurclic_redirection.php?urlid=3** (pour accéder à l'URL ayant l'identifiant 3). À supposer que l'URL 3 corresponde au site **http://www.sqlfacile.com**, cela signifie que le lien `<a href="http://www.sqlfacile.com">`devra être remplacé par `<a href="compteurclic_redirection.php?urlid=3>` si l'on souhaite en compter le nombre de clics.

Concrètement, cette table pourra être créée et alimentée avec la fonction `CC_initialiseBD()` du script suivant :

Listing 10.96 : compteurclic_bd_inc.php (début)

```
<?php
    //-----------------------------------
    // Charge les paramètres de connexion
    // à la base de données.
    //-----------------------------------
    require_once("parametres_bd_inc.php");

    /**
     * Fonction de connexion à une base de données
     * s'appuie sur les paramètres fournis
     * dans parametres_bd_inc.php.
     *
     * @return resource Identifiant de connexion
     */
    function CC_connexion()
    {
        global $serveur, $base, $utilisateur, $motDePasse;

        $idConnexion = mssql_pconnect($serveur, $utilisateur, $motDePasse);
        if (!$idConnexion) return FALSE;
        mssql_select_db($base);

        return $idConnexion;
    }

    /**
     * Fonction de deconnexion.
     * (Ne fait rien sachant que la fonction
     * de connexion établit une connexion persistante)
     */
    function CC_deconnexion($idConnexion)
    {
        return TRUE;
    }

    /**
     * Fonction chargé de créer et d'alimenter
```

```
     * la table "compteur de clics"
     **/
    function CC_initialiseBD($idConnexion, $table)
    {
        // Création de la table
        $requete = "DROP TABLE $table";
        @mssql_query($requete, $idConnexion);

        $requete = "CREATE TABLE $table (".
                                    "urlId INTEGER ".
                                          "IDENTITY PRIMARY KEY,".
                                    "url VARCHAR(128) NOT NULL,".
                                    "nbclic INTEGER DEFAULT 0,".
                                    "UNIQUE(url))";
        if (!mssql_query($requete, $idConnexion)) return FALSE;

        // Alimentation de la table
        $requete = "INSERT INTO $table (url)".
                        " VALUES ('http://www.php.net')";
        if (!mssql_query($requete, $idConnexion)) return FALSE;

        $requete = "INSERT INTO $table (url)".
                        " VALUES ('http://www.phpfacile.com')";
        if (!mssql_query($requete, $idConnexion)) return FALSE;

        $requete = "INSERT INTO $table (url)".
                        " VALUES ('http://www.sqlfacile.com')";
        if (!mssql_query($requete, $idConnexion)) return FALSE;

        $requete = "INSERT INTO $table (url)".
                        " VALUES ('http://www.xmlfacile.com')";
        if (!mssql_query($requete, $idConnexion)) return FALSE;

        $requete = "INSERT INTO $table (url)".
                        " VALUES ('http://www.ootoogo.com')";
        if (!mssql_query($requete, $idConnexion)) return FALSE;

        return TRUE;
    }
?>
```

La lecture de la liste des liens disponibles en base de données pourra se faire via la fonction `CC_recupereLiens()` du script suivant :

Listing 10.97 : compteurclic_bd_inc.php (milieu)

```
<?php
    /**
     * Fonction retournant les informations de liens
     * sous forme d'un tableau associatif possedants
     * les clés
     * - "lien" associé au tableau des liens hypertextes
```

```
     * - "nbclic" associé au tableau des nombres de clics
     **/
    function CC_recupereLiens($idConnexion, $table)
    {
        // Nom du script chargé du comptage et de la redirection
        $script = "compteurclic_redirection.php";

        // Requête SELECT
        $requete = "SELECT * FROM $table";
        $idResultat = mssql_query($requete, $idConnexion);

        // Récupération des enregistrements les uns après les autres
        while ($enreg = mssql_fetch_array($idResultat)) {
            $liens["lien"][] = "<a href=\"$script?urlid=".
                               $enreg["urlId"]."\">".
                               $enreg["url"]."</a>";
            $liens["nbclic"][] = $enreg["nbclic"];
        }

        return $liens;
    }
?>
```

L'affichage des liens consiste uniquement à mettre en page le contenu du tableau ainsi récupéré (voir la fonction du script *compteurclic_html_inc.php*).

Comme cela a été précisé, le lien `<a href="http://www.sqlfacile.com">http://www.sqlfacile.com</a>` a été remplacé par `<a href="compteurclic_redirection.php?urlid=3">http://www.sqlfacile.com </a>`. La redirection vers le site **www.sqlfacile.com** est donc à la charge du script *compteurclic_redirection.php*.

Listing 10.98 : compteurclic_redirection.php

```
<?php

    // Paramètres du script
    require_once("compteurclic_bd_inc.php");
    $table = "compteurclic";

    // Connexion à la base de données
    $idConnexion = @CC_connexion();
    if (!$idConnexion) {
        die("<b>Impossible de se connecter à la base de données</b>");
    }

    // Récupération de l'url et incrémentation du compteur
    // pour l'url d'indentifiant passé en paramètre de ce script
    // par la méthode GET
    $url = @CC_recupereUrl($idConnexion, $_GET["urlid"]);

    // Deconnexion
    @CC_deconnexion();
```

```
    // C'est l'heure de la redirection
    if ($url) {
        header("Location: $url");
    } else {
        echo "Désolé, nous ne pouvons vous proposer ce lien";
    }
?>
```

Ce script appelle principalement la fonction `CC_recupereUrl()` suivante :

Listing 10.99 : compteurclic_bd_inc.php (fin)

```
<?php
    /**
     * Fonction récupérant une url à partir de son identifiant
     * et incrémentant le compteur de clics
     **/
    function CC_recupereUrl($idConnexion, $urlid)
    {
        global $table;

        // Récupère l'url
        $requete = "SELECT * FROM $table WHERE urlid=$urlid";
        $idResultat = mssql_query($requete, $idConnexion);

        if ($enreg = mssql_fetch_array($idResultat)) {
            $url = $enreg["url"];

            // Incrémente le compteur
            $requete = "UPDATE $table SET nbclic=nbclic+1 WHERE urlid=$urlid";
            mssql_query($requete, $idConnexion);
        } else {
            $url = FALSE;
        }
        return $url;
    }
?>
```

Utilisation du bouton retour
Si, après avoir suivi un lien, vous revenez sur la page compteurclic.php en utilisant le bouton retour (back), le contenu de la page ne sera pas réactualisé. Par conséquent, le nombre de visites affiché ne sera pas correct. N'oubliez donc pas, dans ce cas, de rafraîchir (bouton actualiser) la page.

SuperTheque

L'essentiel du code de l'application a été présenté en introduction de ce chapitre. Il ne restait plus qu'a connaître les fonctions MS SQLServer pour implémenter l'interface `RessourceInterface` c'est désormais chose faite:

Listing 10.100 : RessourceMSSQLServer_class.php

```
<?php
include_once("RessourceInterface_class.php");
include_once(dirname(__FILE__)."/../config/mssql_cfg.php");
/**
 * RessourceMSSQLServer_class.php
 * Classe d'acces a une base de donnees MS SQL Server
 * Cette classe doit implementer toutes les methodes de l'interface
 *   RessourceInterface.
 * Compatibilite: PHP 5
 */
class Ressource implements RessourceInterface
{
    var $idConnexion;

    public function connexion()
    {
        global $mssql_serveur, $mssql_utilisateur, $mssql_motDePasse;
        global $mssql_base;

        if (!isset($this->idConnexion)) {
            $idConnexion = mssql_pconnect($mssql_serveur,
                                          $mssql_utilisateur,
                                          $mssql_motDePasse);
            if (!$idConnexion) return FALSE;
            mssql_select_db($mssql_base);
            $this->idConnexion = $idConnexion;
        }
        return TRUE;
    }

    public function deconnexion()
    {
        // Rien a faire, il s'agit d'une connexion persistante
    }

    public function reset()
    {
        $requete = "DROP TABLE types";
        $idResultat = @mssql_query($requete, $this->idConnexion);
        //if (!$idResultat) return FALSE;

        $requete = "CREATE TABLE types (".
                        "id INTEGER IDENTITY PRIMARY KEY,".
                        "type VARCHAR(255))";
```

```
    $idResultat = mssql_query($requete, $this->idConnexion);
    if (!$idResultat) return FALSE;

    $types = array("Album", "Film", "Livre", "Musique");
    foreach ($types as $type) {
        $requete = "INSERT INTO types (type) VALUES ('$type')";
        $idResultat = mssql_query($requete, $this->idConnexion);
        if (!$idResultat) return FALSE;
}

    $requete = "DROP TABLE articles";
    $idResultat = @mssql_query($requete, $this->idConnexion);

    $requete = "CREATE TABLE articles (".
                "id INTEGER IDENTITY PRIMARY KEY,".
                "albumId INTEGER,".
                "titre VARCHAR(255),".
                "typeId INTEGER,".
                "commentaire VARCHAR(255) NULL)";
    $idResultat = mssql_query($requete, $this->idConnexion);
    if (!$idResultat) return FALSE;
}

public function addArticle($albumId,
                           $titre,
                           $typeId,
                           $commentaire)
{
    $requeteDebut = "INSERT INTO articles (albumId, titre, typeId";
    $requeteFin = ") VALUES ($albumId,".
                            "'".addSlashes($titre)."',".
                            "$typeId";
    if ($commentaire != "") {
         $requeteDebut .= ",commentaire";
         $requeteFin .= ",'".addSlashes($commentaire)."'";
    }
    $requete = $requeteDebut . $requeteFin . ")";
    $idResultat = mssql_query($requete, $this->idConnexion);
    if (!$idResultat) return FALSE;
}

public function getArticle($articleId)
{
    $requete = "SELECT * FROM articles WHERE id=$articleId";
    $idResultat = mssql_query($requete, $this->idConnexion);
    if (!$idResultat) return FALSE;

    $article = NULL;
    if ($enreg = mssql_fetch_array($idResultat)) {
        $article = $this->enreg2Article($enreg);
    }
    return $article;
```

```
}

public function getArticles(Filtre $filtre, Plage $plage, Tri $tri)
{
    $nbTotalArticles = $this->getNbTotalArticles($filtre);

    if ($filtre->getAlbumId() != -1) {
        $conditionSQL = "albumId=".$filtre->getAlbumId();
    }
    if ($filtre->getTitre() != "") {
        $conditionSQL = $conditionSQL.
                        " AND titre LIKE '".
                        addSlashes($filtre->getTitre())."'";
    }
    if ($filtre->getTypeId() != -1) {
        $conditionSQL = $conditionSQL.
                        " AND typeId=".$filtre->getTypeId();
    }

    if ($tri->getSens() == -1) {
        $triSQL = "ORDER BY ".$tri->getChamp()." DESC";
        $triInverseSQL = "ORDER BY ".$tri->getChamp()." ASC";
    } else if ($tri->getSens() == 1) {
        $triSQL = "ORDER BY ".$tri->getChamp()." ASC";
        $triInverseSQL = "ORDER BY ".$tri->getChamp()." DESC";
    }

    if ($plage->getPremierArticle()+$plage->getNbArticleMax()
        < $nbTotalArticles) {
        $requete = "SELECT * FROM (SELECT TOP ".
                   $plage->getNbArticleMax().
                   " * FROM (SELECT TOP ".
                   ($plage->getPremierArticle()+
                    $plage->getNbArticleMax()).
                   " * FROM articles";

        $requete = $requete." WHERE ".$conditionSQL." ".

                                    $triSQL.")".
                   " AS filtre1 ".$triInverseSQL.")".
                   " AS filtre2 ".$triSQL;
    } else {
        $requete = "SELECT * FROM (SELECT TOP ".
                   ($nbTotalArticles - $plage->getPremierArticle()).
                   " * FROM articles";

        $requete = $requete." WHERE ".$conditionSQL." ".
                                   $triInverseSQL.")".
                   " AS filtre1 ".$triSQL;
}
    $idResultat = mssql_query($requete, $this->idConnexion);
    if (!$idResultat) return FALSE;
```

```
    $articles = NULL;
    while ($enreg = mssql_fetch_array($idResultat)) {
        $articles[] = $this->enreg2Article($enreg);
    }
    return $articles;
}

public function getNbTotalArticles(Filtre $filtre)
{
    $requete = "SELECT COUNT(*) FROM articles";
    if ($filtre->getAlbumId() != -1) {
        $conditionSQL = "albumId=".$filtre->getAlbumId();
    }
    if ($filtre->getTitre() != "") {
        $conditionSQL = $conditionSQL.
                        " AND titre LIKE '".
                        addSlashes($filtre->getTitre())."'";
    }
    if ($filtre->getTypeId() != -1) {
        $conditionSQL = $conditionSQL.
                        " AND typeId=".$filtre->getTypeId();
    }

    $requete = $requete." WHERE ".$conditionSQL;
    $idResultat = mssql_query($requete, $this->idConnexion);
    if (!$idResultat) return FALSE;

    if ($enreg = mssql_fetch_array($idResultat)) {
        return $enreg[0];
    } else return FALSE;
}


public function getTypes()
{
    $requete = "SELECT * FROM types";
    $idResultat = mssql_query($requete, $this->idConnexion);
    if (!$idResultat) return FALSE;

    $types = NULL;
    while ($enreg = mssql_fetch_array($idResultat)) {
        $types[$enreg["id"]] = $enreg["type"];
    }
    return $types;
}

public function getAlbumTypeId()
{
     return 1;
}
```

```
    private function enreg2Article($enreg)
    {
        $article = new Article();
        $article->setId($enreg["id"]);
        $article->setAlbumId($enreg["albumId"]);
        $article->setTitre($enreg["titre"]);
        $article->setTypeId($enreg["typeId"]);
        $article->setCommentaire($enreg["commentaire"]);
        return $article;
    }

}
?>
```

Comme vous le constatez, la principale difficulté n'est pas tant au niveau des fonctions disponibles qu'au niveau de la construction des requêtes SQL. La requête sera simple lorsqu'il ne s'agit que de récupérer un enregistrement identifié par la clé `Id` (comme c'est le cas avec `getArticle()`) elle sera bien plus complexe s'il s'agit des récupérer un ensemble d'enregistrements répondant à des critères précis et triés (comme c'est le cas avec `getArticles()`).

Requêtes avec des apostrophes

Lors de la construction de la requête, il faut bien garder à l'esprit que l'un des éléments de la requête (typiquement un champ de type texte saisi par l'utilisateur, comme ici le titre ou le commentaire) peut contenir une apostrophe qui pourrait être confondu avec le délimiteur de chaîne. Pensez donc bien à faire un appel à `addSlashes()` lorsque cela peut s'avérer nécessaire.

Affichage par page

Contrairement à d'autres bases de données, Microsoft SQL Server ne dispose pas d'instruction permettant de récupérer `N` enregistrements à partir de l'enregistrement d'index `I`. Il est par contre possible de ne récupérer que les `N` premiers enregistrements grâce à l'instruction `TOP`.

Il est donc possible de récupérer les `I + N` premiers articles puis d'aller directement à l'article `I` grâce à la fonction PHP `mssql_data_seek()`, qui permet de modifier le pointeur de sorte que le premier appel à une fonction telle que `mssql_fetch_row()` ne renvoie pas la première ligne, mais la `I`-ième.

mssql_data_seek()

Permet de déplacer le pointeur pour un appel à `mssql_fetch_row()`, `mssql_fetch_array()` ou encore `mssql_fetch_object()`.

Syntaxe `boolean mssql_data_seek(resource $idResultat, int $numeroLigne)`

`$idResultat`	Identifiant de résultat tel que retourné par `mssql_query()`.
`$numeroLigne`	Numéro de la ligne sur laquelle il faut déplacer le pointeur.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

Si le nombre d'enregistrements dans la base est vraiment important (et que l'on recherche les données correspondant à un numéro de page élevé), alors l'utilisation de TOP n'apportera finalement que peu de choses (puisque, de toute façon, il y aura bien plus de `N` enregistrements retournés). Il peut alors être intéressant de demander à la base de données de ne retourner que les N enregistrements à partir de celui d'index `I`, quitte à lui donner plus de travail avec selon les cas, une requête:

```
SELECT * FROM (SELECT TOP <N> * FROM (SELECT TOP <I+N> * FROM table ORDER BY champ) AS filtre1 ORDER BY champ DESC) AS filtre2 ORDER BY champ.
```

si I+N est inférieur au nombre total d'enregistrements répondant aux critères de sélection ou à la requête:

```
SELECT * FROM (SELECT TOP <nbTotalArticles - I> * FROM table ORDER BY champ DESC) AS filtre1 ORDER BY champ.
```

dans le cas contraire.

Tri

Modifier l'ordre d'affichage des annonces n'est pas non plus bien sorcier. Le tri peut s'effectuer directement au niveau de la requête SQL grâce à l'instruction `ORDER BY`.

Moteur de recherche (filtre)

Pour filtrer, il suffit d'utiliser l'instruction `WHERE`. Dans notre cas, nous souhaitons autoriser l'utilisation de jokers (comme %) afin, par exemple, de rechercher les titres commençant par une chaîne donnée. C'est pourquoi, nous ne ferons pas un test de type `titre='motcle'` mais `titre LIKE 'motcle'` (où `motcle` pourrait avoir la valeur "La 7eme compagnie%").

En savoir plus...

... sur le résultat d'une requête

Il vous est possible de connaître le nombre de champs retournés par une requête (et par conséquent le nombre de champs d'une table dans le cas d'une requête `SELECT * FROM <nom_table>` sur une table non vide).

mssql_num_fields()

Retourne le nombre de champs du résultat.

Syntaxe `int mssql_num_fields(resource $resultat)`

`$resultat`	Identifiant de résultat tel que retourné par `mssql_query()`.
`retour`	Nombre de champs disponibles dans le résultat.

mssql_fetch_field()

Retourne un objet contenant des informations sur un champ.

Syntaxe	`object mssql_fetch_field(resource $resultat [, int $numeroColonne])`
`$resultat`	Identifiant de résultat tel que retourné par `mssql_query()`.
`$numeroColonne`	Numéro de la colonne dont on veut des informations. Si ce paramètre n'est pas passé, les informations obtenues sont celles qui n'ont pas été récupérées par `mssql_fetch_field()`.
`retour`	Un objet possédant les attributs : `name` : nom de la colonne ; si cette colonne est le fruit d'une fonction, alors le nom retourné sera `computed` suivi d'un nombre. `column_source` : table d'où provient la colonne. `max_length` : taille maximale de la colonne. `numeric` : `1` si la colonne est une colonne contenant des valeurs numériques.

mssql_field_length()

Retourne la taille d'un champ.

Syntaxe	`int mssql_field_length(resource $resultat [, int $numeroChamp])`
`$resultat`	Identifiant de résultat tel que retourné par `mssql_query()`.
`$numeroChamp`	Numéro du champ sur lequel vous souhaitez des informations. Si ce paramètre n'est pas passé, les informations obtenues sont celles qui n'ont pas été récupérées par `mssql_fetch_field()`.
`retour`	La taille du champ.

mssql_field_name()

Retourne le nom d'un champ.

Syntaxe	`string mssql_field_name(resource $resultat [, int $numeroChamp])`

`$resultat`	Identifiant de résultat tel que retourné par `mssql_query()`.
`$numeroChamp`	Numéro du champ dont on veut des informations. Si ce paramètre n'est pas passé, les informations obtenues sont celles qui n'ont pas été récupérées par `mssql_fetch_field()`.
`retour`	Le nom du champ.

mssql_field_type()

Retourne le type d'un champ.

Syntaxe	`string mssql_field_name(resource $resultat [, int $numeroChamp])`
`$resultat`	Identifiant de résultat tel que retourné par `mssql_query()`.
`$numeroChamp`	Numéro du champ dont on veut des informations. Si ce paramètre n'est pas passé, les informations obtenues sont celles qui n'ont pas été récupérées par `mssql_fetch_field()`.
`retour`	Le type du champ.

mssql_field_seek()

Déplace le pointeur de champ vers un autre.

Syntaxe	`boolean mssql_field_seek(resource $resultat, int $numeroColonne)`
`$resultat`	Identifiant de résultat tel que retourné par `mssql_query()`.
`$numeroColonne`	Numéro de la colonne où déplacer le pointeur.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

10.14. Les couches d'abstraction

Comme le démontre le sous-chapitre précédent et la liste de bases de données associée, il existe, quasiment, une bibliothèque spécifique pour chaque serveur, alors que, quelle que soit la base, la nature des opérations à réaliser est toujours du même type (connexion, soumission d'une requête, lecture des résultats, etc.).

Tenant compte de cette observation, d'aucuns proposent des bibliothèques fournissant des fonctions identiques quelle que soit la base de données interrogée (on parle alors de couche d'abstraction). C'est du moins l'objectif, car, généralement, dans les faits, seul un petit nombre de bases de données est supporté.

L'intérêt des couches d'abstraction est bien évidemment de permettre de réaliser facilement un script pouvant s'adapter à n'importe quelle base de données. Mais, là encore, il s'agit bien souvent d'un vœu pieu. En effet, comme vous avez pu le constater si vous avez lu le chapitre *ODBC*, toutes les bases de données n'implémentent pas les mêmes instructions SQL, et chacune propose sa propre version. Ainsi, même avec une couche d'abstraction, le code doit être adapté et testé pour chaque type de base de données. Cela ne rend toutefois pas totalement inutiles les couches d'abstraction.

Parmi les couches d'abstraction existantes nous avons :

- La bibliothèque `Dbx` ;
- La bibliothèque `PEAR`.

  Nous ne traiterons ici que la couche d'abstraction PEAR qui est la plus complète et nous semble par conséquent la plus intéressante.

Pear DB

La désormais célèbre bibliothèque `PEAR` propose également une couche d'abstraction pour les bases de données. Celle-ci supporte à l'heure actuelle (avec plus ou moins de bonheur) les bases de données MySQL, PostgresSQL, Interbase, MiniSQL, MS SQL Server, Oracle 8i, Sybase, Informix, Frontbase, ainsi que les liaisons ODBC.

Installation

L'utilisation de `PEAR DB` ne nécessite aucune installation spécifique (hormis l'installation des bases de données que vous souhaitez utiliser et leur "intégration" à PHP). Il suffit d'activer le support `PEAR` au niveau de PHP.


RENVOI

Vous pouvez vous reporter aux sections propres aux bases de données pour plus de détails sur la façon de faire pour les "intégrer" à PHP.

Configuration de PHP avec le support PEAR

Si vous ne disposez pas du paquetage DB ou si vous souhaitez obtenir la dernière version disponible tapez simplement (dans une console) la commande suivante

```
pear upgrade DB
```

Chaque script nécessitant l'utilisation de `PEAR DB` devra inclure la ligne :

```
require_once("DB.php");
```

Utilisation

L'utilisation basique de la base de données s'opère selon le schéma suivant :

- Connexion grâce à la méthode statique de `DB::connect()`.

- Soumission de la requête via la méthode `DB->query()`.
- Déconnexion grâce à la méthode `DB->disconnect()` (dans le cas d'une connexion non persistante).

Connexion

La connexion à la base de données s'opère grâce à la méthode statique `DB::connect()`.

DB::connect()

Établit une connexion (persistante ou non) avec le serveur de bases de données.

Syntaxe	`object DB::connect(string $baseDeDonnees [, boolean $persistance])`
`$baseDeDonnees`	Précise la base de données selon la syntaxe `typeServeur [(syntaxeSQL)]://[utilisateur[:motDePasse]@] [protocole()[serveur][)][/baseDeDonnees]`, où : `typeServeur` peut prendre l'une des valeurs suivantes "`fbsql`" pour `Frontbase`, "`ibase`" pour Interbase, "`ifx`" pour Informix, "`mysql`" pour MySQL, "`msql`" pour MiniSQL, "`mssql`" pour MS SQL Server, "`oci8`" pour Oracle 8i, "`odbc`" pour une liaison ODBC, "`pgsql`" pour PostgreSQL ou "`sybase`" pour Sybase. `syntaxeSQL` peut prendre les valeurs "`dbase`", " `fbsql`", "`ibase`", "`firebird`", "`ifx`", "`msql`", "`mssql`", "`mysql`", "`oci8`", "`sql92`", "`pgsql`", "`sybase`". `utilisateur` est le nom de l'utilisateur. `motDePasse` est le mot de passe de l'utilisateur. `protocole` peut prendre les valeurs "`tcp`" ou "`unix`" (socket). `serveur` est nom du serveur éventuellement suivi de `:` puis du numéro du port (protocole "`tcp`") ou le fichier de socket (protocole "`unix`") `baseDeDonnees` est le nom de la base de données, ou éventuellement le nom et le chemin du fichier de base de données.
`$persistance`	Indiquez `TRUE` si vous souhaitez une connexion persistante, `FALSE` (valeur par défaut) sinon.
`retour`	Objet *DB* en cas de succès, objet *DB_Error* sinon.

Exécution de la requête SQL

L'exécution d'une requête SQL s'opère par un simple appel à la méthode `DB->query()`.

DB->query()

Exécute une requête SQL.

Syntaxe	`mixed query(string $requete)`
`$requete`	Requête SQL.
`retour`	Dans le cas d'une requête ne retournant pas de résultat (ex. : `CREATE`, `INSERT`, `UPDATE`), la méthode retourne la constante `DB_OK` (1) en cas de succès, ou un objet *DB_Error* sinon. Dans le cas d'une requête retournant des résultats (ex : `SELECT`) la méthode retourne un objet *DB_Result* en cas de succès, ou un objet *DB_Error* sinon.

Une des fonctions les plus intéressantes de cette couche d'abstraction est probablement la suivante:

DB->limitQuery()

Exécute une requête SQL en ne demandant qu'un sous-ensemble des résultats (à la manière de l'instruction `LIMIT` que l'on peut retrouver avec MySQL et SQLite).

Syntaxe	`mixed limitQuery(string $requete, int $premierEnregistrement, int $nbEnregistrements)`
`$requete`	Requête SQL.
`$premier Enregistrement`	Index du premier enregistrement à retourner.
`$nbEnregistrements`	Nombre d'enregistrement maximum à retourner
`retour`	Dans le cas d'une requête ne retournant pas de résultat (ex. : `CREATE`, `INSERT`, `UPDATE`), la méthode retourne la constante `DB_OK` (1) en cas de succès, ou un objet *DB_Error* sinon. Dans le cas d'une requête retournant des résultats (ex : `SELECT`) la méthode retourne un objet *DB_Result* en cas de succès, ou un objet *DB_Error* sinon.

Nous verrons par la suite que, pour les bases de données le supportant, il est également possible d'utiliser des requêtes préparées (contenant des paramètres qui pourront être modifiés juste avant que celles-ci ne soient exécutées), et de désactiver le mode `auto commit` (validation automatique) pour avoir accès aux opérations de `commit` (validation) et `rollback` (annulation).

Dans le cas d'une requête retournant un résultat (typiquement une requête `SELECT`), il faudra également analyser le résultat. Mais nous verrons cela un peu plus loin.

Déconnexion

Pour vous déconnecter – opération théoriquement facultative mais toutefois vivement conseillée –, vous disposez de la méthode `DB->disconnect()`.

DB->disconnect()

Met fin à la connexion à la base de données et libère les ressources associées.

Syntaxe	`boolean disconnect()`
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

Premier exemple

Nous voilà donc à même de construire notre premier script utilisant une base de données.


ATTENTION

Vérifiez les paramètres
Prenez garde ! Avant d'exécuter ce script, assurez-vous que les paramètres prédéfinis ne risquent pas de conduire à la suppression des données d'une de vos bases qui, par coïncidence, existerait déjà.

Comme cela est fortement conseillé, nous avons, ici, isolé les paramètres de connexion dans un fichier aisément repérable.

Listing 10.101 : parametres_bd_inc.php

```
<?php

    // Paramètres de connexion à la base de données
    // N'hésitez pas à les modifier selon vos besoins

    // $typeServeur peut prendre l'une des valeurs suivante
    // "mysql"    - MySQL
    // "pgsql"    - PostgreSQL
    // "oci8"     - Oracle 8i
    // "MSAccess" - MS Access
    // "IBMDB2"   - IBM DB2
    // ...

    $typeServeur = "mysql";
    $serveur     = "localhost";
    $base        = "mabase";
    $utilisateur = "root";
    $motDePasse  = "";
?>
```

Notre script principal sera donc :

Listing 10.102 : insert01.php

```
<?php

    // Paramètres du script
```

```
    require_once("parametres_bd_inc.php");
    $table = "tableexemple";

    // Inclusion du script contenant les fonctions
    require_once("insert01_bd_inc.php");

    // Connexion à la base de données
    switch ($typeServeur) {
        case "MSAccess" :
            $typeServeurPear = "odbc";
            break;
        default :
            $typeServeurPear = $typeServeur;
    }

    $dsn = "$typeServeurPear://";
    if (!empty($utilisateur)) $dsn .= $utilisateur;
    if (!empty($motDePasse))  $dsn .= ":$motDePasse";
    $dsn .= "@tcp($serveur)";
    if (!empty($base)) $dsn .= "/$base";

    $bd = DB::connect($dsn, TRUE);
    if (DB::isError($bd)) {
        echo $dsn." ".$bd->message;
        return FALSE;
    }
    if ($bd === FALSE) {
        die("<b>Impossible de se connecter à la base de données</b>");
    }

    // Appel de la fonction principale
    if (EX_initialiseBD($bd, $table)) {
        echo "Voilà, une nouvelle table avec quelques données, ".
             "vous pouvez le vérifier avec votre client de base ".
             " de données ou via les scripts suivants.";
    } else {
        echo "La création ou l'alimentation de la table à échouée.";
    }

    // Pas de déconnexion dans le cas d'une connexion persistante
    // $db->disconnect();
?>
```

et nécessitera :

Listing 10.103 : insert01_bd_inc.php

```
<?php
   require_once("DB.php");

    /**
```

```
     * Fonction chargée de créer et d'alimenter une table
     *
     * @param $bd   object Base de données
     * @param $table       string   Nom de la table
     **/
    function EX_initialiseBD($bd, $table)
    {
        // Supprime la précédente table
        $requete = "DROP TABLE $table";
        @$bd->query($requete);

        // Crée la table
        $requete = "CREATE TABLE $table (filmId INTEGER, ".
                                        "film VARCHAR(64))";
        $sts = $bd->query($requete);
        if (DB::isError($sts)) return FALSE;

        // Ajoute quelques données
        $requete = "INSERT INTO $table VALUES (1, 'Forrest Gump')";
        $sts = $bd->query($requete);
        if (DB::isError($sts)) return FALSE;

        $requete = "INSERT INTO $table VALUES (2, 'Matrix')";
        $sts = $bd->query($requete);
        if (DB::isError($sts)) return FALSE;

        $requete = "INSERT INTO $table VALUES (3, 'La cité de la peur')";
        $sts = $bd->query($requete);
        if (DB::isError($sts)) return FALSE;

        return TRUE;
    }
?>
```

Commit/rollback

Par défaut, les requêtes sont automatiquement validées (mode `auto commit`). Il est toutefois possible, pour certaines bases de données, de modifier ce comportement en utilisant la méthode `DB->autoCommit()`.

DB->autoCommit()

Active ou désactive le mode de validation automatique des transactions (`auto commit`).

Syntaxe	`mixed autoCommit([boolean $mode])`
`$mode`	`TRUE` pour activer le mode `auto commit`, `FALSE` (valeur par défaut) sinon.

`retour`	`DB_OK` si la base de données supporte les transactions, ou l'objet *DB_Error* sinon.

Si vous avez opté pour un mode de validation non automatique, vous pourrez valider ou annuler vos transactions avec les fonctions suivantes :

DB->commit()

Valide la transaction en cours.

Syntaxe	`mixed commit(void)`
`retour`	`DB_OK` si la base de données supporte les transactions, ou l'objet *DB_Error* sinon.

DB->rollback()

Annule la transaction en cours.

Syntaxe	`mixed rollback(void)`
`retour`	`DB_OK` si la base de données supporte les transactions, ou l'objet *DB_Error* sinon.

Lecture des enregistrements

Dans le cas d'une requête retournant une liste de données (typiquement un `SELECT`), il convient de récupérer l'objet *DB_Result* et de l'utiliser pour lire, en boucle, ligne après ligne, chaque enregistrement. Il existe pour cela deux méthodes au comportement similaire, mais avec des syntaxes différentes.

Nous verrons ensuite qu'il est également possible d'exécuter des requêtes et de récupérer les enregistrements sans même passer par un objet *DB_Result*.

Lecture enregistrement par enregistrement

DB_Result->fetchRow()

Retourne l'enregistrement courant et déplace le curseur vers l'enregistrement suivant.

Syntaxe	`mixed fetchRow([int $mode [,int $enregistrementIdx]])`
`$mode`	Détermine le format de la réponse, cela peut prendre l'une des valeurs suivantes :

	DB_FETCHMODE_ASSOC pour restituer l'enregistrement sous la forme d'un tableau associatif où les clés sont les noms de champs. DB_FETCHMODE_OBJECT pour restituer l'enregistrement sous la forme d'un objet. DB_FETCHMODE_ORDERED pour restituer l'enregistrement sous la forme d'un tableau indexé (commençant à 0) et contenant les valeurs des champs (dans l'ordre de la requête). DB_FETCHMODE_DEFAULT (valeur par défaut si la méthode setFetchMode() n'a pas été appelée) pour restituer l'enregistrement sous sa forme par défaut (probablement DB_FETCHMODE_ORDERED).
$enregistrementIdx	Index de l'enregistrement à restituer.
retour	Selon le mode choisi, en cas de succès : un tableau indexé, associatif, un objet, ou NULL s'il n'y a plus d'enregistrement à lire.

La fonction suivante listera donc le contenu de la table :

Listing 10.104 : select_eea_bd_inc.php

```
<?php
    require_once("DB.php");

    /**
     * Fonction listant le contenu d'une table
     * contenant 2 champs (filmId et film)
     *
     * @param $bd   object Base de données
     * @param $table       string   Nom de la table
     **/
    function EX_listeContenu($bd, $table)
    {
        // Requete
        $requete = "SELECT * FROM $table";
        $resultat = $bd->query($requete);
        if (DB::isError($resultat)) return FALSE;

        // Boucle de lecture (et d'affichage) des enregistrements
        while ($enreg = $resultat->fetchRow(DB_FETCHMODE_ASSOC)) {
            echo "FilmId=".$enreg["filmId"]." ".
                 "Film =" .$enreg["film"]."<br />";
        }
        return TRUE;
     }
?>
```

L'autre méthode similaire est DB_Result->fetchInto.

DB_Result->fetchInto()

Retourne l'enregistrement courant et déplace le curseur vers l'enregistrement suivant.

Syntaxe	`int fetchInto(array &$enregistrement [, int $mode [, int $enregistrementIdx]])`
`$enregistrement`	Variable dans laquelle copier l'enregistrement.
`$mode`	Détermine le format de la réponse, cela peut prendre l'une des valeurs suivantes : `DB_FETCHMODE_ASSOC` pour restituer l'enregistrement sous la forme d'un tableau associatif où les clés sont les noms de champs. `DB_FETCHMODE_OBJECT` pour restituer l'enregistrement sous la forme d'un objet. `DB_FETCHMODE_ORDERED` pour restituer l'enregistrement sous la forme d'un tableau indexé (commençant à 0) et contenant les valeurs des champs (dans l'ordre de la requête). `DB_FETCHMODE_DEFAULT` (valeur par défaut si la méthode `setFetchMode()` n'a pas été appelée) pour restituer l'enregistrement sous sa forme par défaut (probablement `DB_FETCHMODE_ORDERED`).
`$enregistrementIdx`	Index de l'enregistrement à restituer.
`retour`	La constante `DB_OK` (1) en cas de succès, `FALSE` sinon.

Sans objet DB_Result

L'objet *DB* de la bibliothèque `PEAR` propose une série de méthodes permettant d'accéder directement aux résultats d'une requête, et répondant aux cas d'utilisation les plus fréquents.

DB->getOne()

Retourne le premier champ du premier enregistrement retourné par une requête (ex. : du type `SELECT champ FROM table WHERE id=valeur`).

Syntaxe	`string getOne(mixed $requete [, array $parametres])`
`$requete`	Requête SQL, ou identifiant d'une requête préparée tel que retourné par `DB->prepare()`.
`$parametres`	Paramètres de la requête préparée.
`retour`	La valeur du champ, ou un objet *DB_Error* en cas d'erreur.


RENVOI

Voir l'exemple d'utilisation dans le paragraphe "Compter les enregistrements".

DB->getRow()

Retourne le premier enregistrement retourné par une requête (ex. : du type `SELECT * FROM table WHERE id=valeur`).

Syntaxe	`mixed getRow(mixed $requête [, array $parametres [, int $mode]])`
`$requete`	Requête SQL, ou identifiant d'une requête préparée tel que retourné par `DB->prepare()`.
`$parametres`	Paramètres de la requête préparée (`NULL` si vous souhaitez simplement préciser le mode).
`$mode`	Détermine le format de la réponse, cela peut prendre l'une des valeurs suivantes : `DB_FETCHMODE_ASSOC` pour restituer l'enregistrement sous la forme d'un tableau associatif où les clés sont les noms de champs. `DB_FETCHMODE_OBJECT` pour restituer l'enregistrement sous la forme d'un objet. `DB_FETCHMODE_ORDERED` pour restituer l'enregistrement sous la forme d'un tableau indexé (commençant à 0) et contenant les valeurs des champs (dans l'ordre de la requête). `DB_FETCHMODE_DEFAULT` (valeur par défaut si la méthode `setFetchMode()` n'a pas été appelée) pour restituer l'enregistrement sous sa forme par défaut (probablement `DB_FETCHMODE_ORDERED`).
`retour`	Selon le mode choisi, en cas de succès : un tableau indexé, associatif, un objet, ou `NULL` si la requête ne retourne aucun enregistrement ; voire un objet *DB_Error* en cas d'erreur.

DB->getCol()

Retourne l'ensemble des valeurs d'un champ des enregistrements retournés par une requête.

Syntaxe	`mixed getRow(mixed $requête [, string $champ [, array $parametres]])`
`$requete`	Requête SQL, ou identifiant d'une requête préparée tel que retourné par `DB->prepare()`.
`$champ`	Index ou nom du champ.
`$parametres`	Paramètres de la requête préparée.

`retour`	Tableau indexé contenant les valeurs pour chaque enregistrement retourné, ou un objet *DB_Error* en cas d'erreur.

DB->getAssoc()

Retourne l'ensemble des enregistrements retournés par une requête sous la forme d'un tableau associatif basé sur la valeur du premier champ.

Syntaxe	`array getAssoc(mixed $requete [, boolean $modeTableau [, array $parametres ]])`
`$requete`	Requête SQL, ou identifiant d'une requête préparée tel que retourné par `DB->prepare()`.
`$modeTableau`	`TRUE` pour forcer l'utilisation d'un tableau, même dans les cas où chaque clé du tableau n'est associée qu'à une seule valeur, `FALSE` (valeur par défaut) sinon.
`$parametres`	Paramètres de la requête préparée.
`retour`	Dans le cas d'enregistrements avec un seul champ, retourne la constante `DB_ERROR_TRUNCATED`. Dans le cas d'enregistrements avec deux champs et `$modeTableau` à `FALSE`, retourne un tableau associatif ayant pour clés les valeurs du premier champ et pour valeurs celles du second champ. Dans le cas d'enregistrements avec deux champs et `$modeTableau` à `TRUE`, retourne un tableau associatif ayant pour clés les valeurs du premier champ et pour valeur un tableau contenant les valeurs du second champ correspondant. Dans le cas d'enregistrements avec plus de trois champs, retourne un tableau associatif ayant pour clés les valeurs du premier champ et pour valeur un tableau contenant les valeurs des autres champs correspondants. En cas d'erreur, retourne un objet *DB_Error*. À noter : s'il y a des doublons dans les valeurs du premier champ, seuls les derniers enregistrements associés à ces valeurs doublons sont conservés.

DB->getAll()

Retourne l'ensemble des enregistrements retournés par une requête sous la forme d'un tableau indexé.

Syntaxe	`array getAll(mixed $requete [, array $parametres [, int $mode]])`
`$requete`	Requête SQL, ou identifiant d'une requête préparée tel que retourné par `DB->prepare()`.
`$parametres`	Paramètres de la requête préparée.

`$mode`	Détermine le format de la réponse ; cela peut prendre l'une des valeurs suivantes : `DB_FETCHMODE_ASSOC` pour restituer les enregistrements sous la forme d'un tableau indexé contenant un tableau associatif où les clés sont les noms de champs. `DB_FETCHMODE_ORDERED` pour restituer les enregistrements sous la forme d'un tableau indexé contenant un tableau indexé associé aux valeurs des champs (dans l'ordre de la requête). `DB_FETCHMODE_FLIPPED` combiné par `OU` logique avec les modes ci-dessus permet d'intervertir les rôles des champs et enregistrements dans le tableau. (Avec `DB_FETCHMODE_ASSOC` cela donne un tableau associatif, où les clés sont les noms des champs, contenant un tableau indexé des valeurs pour chaque enregistrement. Avec `DB_FETCHMODE_ORDERED`, cela donne un tableau indexé (où chaque entrée représente un champ, dans l'ordre de la requête) contenant les valeurs pour chaque enregistrement. `DB_FETCHMODE_DEFAULT` (valeur par défaut si la méthode `setFetchMode()` n'a pas été appelée) pour restituer l'enregistrement sous sa forme par défaut (probablement `DB_FETCHMODE_ORDERED`).
`retour`	Un tableau dont la structure dépend du mode choisi, ou un objet *DB_Error* en cas d'erreur.

Nombre d'enregistrements

Nombre d'enregistrements retournés

Il existe trois façons de déterminer le nombre d'enregistrements retournés par une requête.

Si vous souhaitez uniquement connaître le nombre d'enregistrements, mais ne souhaitez pas immédiatement lire ces enregistrements, alors il vous suffit de faire une requête `COUNT()` comme suit :

Listing 10.105 : count_bd_inc.php

```
<?php
    require_once("DB.php");

    /**
     * Fonction affichant le nombre d'enregistrements
     * dans une table
     *
     * @param $bd   object Base de données
     * @param $table       string   Nom de la tabme
     **/
    function EX_compte($bd, $table)
    {
        // Requête
        $requete = "SELECT COUNT(*) FROM $table";
```

```
        $resultat = $bd->getOne($requete);
        if (DB::isError($resultat)) return FALSE;

        echo "Il y a $resultat enregistrements dans la table.<br />";
        return TRUE;
    }
?>
```

Il est également envisageable de les compter au fur et à mesure de leur lecture (s'ils sont lus, par exemple, avec la méthode `getRow()`).

Mais, surtout, ce que vous attendez tous : si vous souhaitez connaître le nombre d'enregistrements avant de les lire, vous pouvez faire appel à la méthode `DB_Result->numRows()`.

DB_Result->numRows()

Indique le nombre d'enregistrements retournés par une requête.

Syntaxe	`int numRows(void)`
`retour`	Nombre d'enregistrements retournés, ou un objet *DB_Error* en cas d'erreur.

Listing 10.106 : count_num_rows_bd_inc.php

```
<?php
    require_once("DB.php");

    /**
     * Fonction affichant le nombre d'enregistrements
     * dans une table
     *
     * @param $bd   object Base de données
     * @param $table       string   Nom de la tabme
     **/
    function EX_compte($bd, $table)
    {
        // Requête
        $requete = "SELECT * FROM $table";
        $resultat = $bd->query($requete);
        if (DB::isError($resultat)) return FALSE;

        echo "Il y a ".$resultat->numRows()." enregistrements".
             " dans la table.<br />";
        return TRUE;
    }
?>
```

Sachant qu'une requête `SELECT *` est plus longue qu'une requête `SELECT COUNT(*)`, privilégiez la première méthode si vous n'avez que faire du contenu des enregistrements.

Nombre d'enregistrements modifiés

Pour déterminer le nombre d'enregistrements modifiés, vous pouvez appeler la méthode `DB->affectedRows()`.

DB->affectedRows()

Indique le nombre d'enregistrements modifiés lors de l'exécution de la requête (de type `INSERT`, `UPDATE`).

Syntaxe	`int affectedRows(void)`
`retour`	Nombre d'enregistrements modifiés ou un objet *DB_Error* en cas d'erreur.

Mise à profit des requêtes préparées

Certains serveurs de bases de données permettent l'analyse, la compilation et le stockage des requêtes avant utilisation. Ceci permet d'exécuter une série de requêtes similaires sans avoir à renouveler à chaque fois les opérations d'analyse et de compilation (mais seulement en changeant certaines valeurs). Même si cela ne s'applique que pour certains serveurs, les méthodes qui suivent restent applicables quelle que soit la base de données (couche d'abstraction oblige). Pour les bases de données ne supportant pas les requêtes préparées, cela sera tout simplement émulé.

Pour cela, il suffit de préparer une requête dans laquelle les éléments variables sont remplacés par des points d'interrogation (ex. : `INSERT INTO matable (film) VALUES (?)`). Il suffira de préciser ces valeurs au moment de son exécution.

La préparation de la requête se fait via la fonction `DB->prepare()`.

DB->prepare()

Prépare une requête SQL.

Syntaxe	`resource prepare(string $requete)`
`$requete`	Requête SQL à préparer.
`retour`	Identifiant de la requête préparée (ce n'est pas une véritable ressource, mais plutôt un index si la base de données ne supporte pas les requêtes préparées).

L'exécution proprement dite est assurée par `DB->execute()`.

DB->execute()

Exécute une requête préparée.

Syntaxe	`object execute(resource $idRequetePreparee, array $parametres)`
`$idRequetePreparee`	Identifiant de la requête préparée.
`$parametres`	Tableau indexé contenant les valeurs des différents paramètres de la requête préparée.
`retour`	La constante `DB_OK` en cas de succès d'une requête ne retournant pas de résultat, un objet `DB_Result` en cas de succès d'une requête retournant un résultat, un objet *DB_Error* sinon.

Mais il est également possible d'enchaîner automatiquement plusieurs appels.

DB->executeMultiple()

Exécute une requête préparée pour l'ensemble des données d'un tableau.

Syntaxe	`object executeMultiple(resource $idRequetePreparee, array $parametres)`
`$idRequetePreparee`	Identifiant de la requête préparée.
`$parametres`	Tableau indexé contenant une entrée par requête à exécuter. Chaque entrée contient un tableau contenant les valeurs des différents paramètres de la requête préparée.
`retour`	La constante `DB_OK` en cas de succès d'une requête ne retournant pas de résultat, un objet `DB_Result` en cas de succès d'une requête retournant un résultat, un objet *DB_Error* sinon.

Un exemple d'utilisation est donné plus loin avec le script *compteurclic*.

Champs auto-incrementés

Une des difficultés rencontrées pour créer des scripts utilisables avec différents serveurs de bases de données concerne les champs auto-incrémentés.

La solution proposée par la bibliothèque `PEAR` consiste à créer une séquence.

DB->createSequence()

Crée une séquence.

Syntaxe	`mixed createSequence(string $nom)`

`$nom`	Nom de la séquence.
`retour`	`DB_OK` en cas de succès, un objet *DB_Error* sinon.

Concrètement, cela se traduit par la création d'une séquence pour les bases de données supportant ce type d'objet ; pour les autres, une table simulant une séquence est créée (ce qui est le cas pour MySQL. La table porte alors le nom indiqué accompagné du suffixe `"_seq"`).

Malheureusement, ceci ne constitue qu'une solution de remplacement aux champs auto-incrémentés. Dès lors, plus rien n'est automatique, et vous devez, avant chaque insertion dans la table, récupérer la valeur suivante de la séquence. Cela s'effectue par un appel à la méthode `nextId()`.

DB->nextId()

Récupère la valeur suivante d'une séquence.

Syntaxe	`int nextId(string $nom)`
`$nom`	Nom de la séquence.
`retour`	Valeur suivante, ou un objet *DB_Error* en cas d'échec.

Vous pouvez également supprimer une séquence.

DB->dropSequence()

Supprime une séquence.

Syntaxe	`mixed dropSequence(string $nom)`
`$nom`	Nom de la séquence à supprimer.
`retour`	`DB_OK` en cas de succès, un objet *DB_Error* sinon.

Un exemple d'utilisation des séquences est donné plus loin avec le script *compteurclic*.

Traitement des chaînes de caractères

DB->escapeSimple()

Prépare une chaîne de caractères pour une insertion dans une requête SQL (par "échappement" des caractères particuliers comme les guillemets ou apostrophes). En d'autres termes il s'agit d'un équivalent des fonctions `mysql_escape_string()`, `sqlite_escape_string()`, etc.

Syntaxe	`string escapeSimple(string $nom)`
`$nom`	Chaîne de caractères à traiter.
`retour`	La chaîne de caractères prête à être insérée dans une requête SQL.

Gestion des erreurs

Afin de déterminer si la valeur retournée par une méthode est un objet *DB_Error* ou non, l'objet *DB* vous propose la méthode `DB::isError()`.

DB::isError()

Teste si l'argument est un objet *DB_Error* ou non.

Syntaxe	`boolean isError(mixed $argument)`
`$argument`	Argument à tester.
`retour`	`TRUE` s'il s'agit d'un objet *DB_Error*, `FALSE` sinon.

Il est cependant possible de déterminer plus précisément l'origine de l'erreur. Il est notamment possible de lire le contenu de l'objet *DB_Error.*

DB_Error est ainsi composé (entre autres) des attributs :

- `level`, niveau d'erreur parmi :
  - 1024 erreur ;
  - `DB_WARNING` (-1 000) avertissement ;
  - `DB_WARNING_READ_ONLY` (-1 001) avertissement.
- `code`, une des valeurs suivantes :
  - `DB_ERROR` (-1) erreur inconnue ;
  - `DB_ERROR_SYNTAX` (-2) la requête comporte une erreur de syntaxe ;
  - `DB_ERROR_CONSTRAINT` (-3) violation de contrainte ;
  - `DB_ERROR_NOT_FOUND` (-4) la base n'existe pas ;
  - `DB_ERROR_ALREADY_EXISTS` (-5) une requête `CREATE` a été demandée pour une table, séquence, etc. existant déjà ;
  - `DB_ERROR_UNSUPPORTED` (-6) ;
  - `DB_ERROR_MISMATCH` (-7) ;
  - `DB_ERROR_INVALID` (-8) ;
  - `DB_ERROR_NOT_CAPABLE` (-9) ;
  - `DB_ERROR_TRUNCATED` (-10) la méthode `DB->getAssoc()` a été appelée alors que le résultat ne comporte qu'un seul champ ;
  - `DB_ERROR_INVALID_NUMBER` (-11) ;

- `DB_ERROR_INVALID_DATE` (-12) ;
- `DB_ERROR_DIVZERO` (-13) division par zéro ;
- `DB_ERROR_NODBSELECTED` (-14) aucune base sélectionnée ;
- `DB_ERROR_CANNOT_CREATE` (-15) ;
- `DB_ERROR_CANNOT_DELETE` (-16) ;
- `DB_ERROR_CANNOT_DROP` (-17) ;
- `DB_ERROR_NOSUCHTABLE` (-18) l'une des tables n'existe pas ;
- `DB_ERROR_NOSUCHFIELD` (-19) l'un des champs n'existe pas ;
- `DB_ERROR_NEED_MORE_DATA` (-20) ;
- `DB_ERROR_NOT_LOCKED` (-21) ;
- `DB_ERROR_VALUE_COUNT_ON_ROW` (-22) ;
- `DB_ERROR_INVALID_DSN` (-23) paramètre `$baseDeDonnees` de `DB->connect()` non valide ;
- `DB_ERROR_CONNECT_FAILED` (-24) ;
- `DB_ERROR_EXTENSION_NOT_FOUND` (-25) ;
- `DB_ERROR_NOSUCHDB` (-25 sic) ;
- `DB_ERROR_ACCESS_VIOLATION` (-26) ;
- `message`, le message d'erreur ;
- `nativecode`, (si disponible) le code d'erreur retourné par la base de données +" `**` "+ le message d'erreur (du moins avec MySQL).

Exemples d'applications

Nous ne reviendrons pas en détail sur les scripts suivants, car ils concernent des exemples présentés dans les chapitres traitant des différentes bases de données. Nous supposons que vous vous êtes penché sur le cas d'au moins une de ces bases et que vous avez, par conséquent, compris le mode de fonctionnement de ces scripts.

Nous nous contenterons donc de vous présenter les corrections à apporter pour l'utilisation de la bibliothèque `Dbx`.

Compteur de clics

Le script *compteurclic_bd_inc.php* deviendra ainsi :

Listing 10.107 : compteurclic_bd_inc.php

```
<?php
    require_once("DB.php");

    //-----------------------------------
    // Charge les paramètres de connexion
    // à la base de données.
    //-----------------------------------
    require_once("parametres_bd_inc.php");
```

```
/**
 * Fonction de connexion à une base de données
 * s'appuie sur les paramètres fournis
 * dans parametres_bd_inc.php.
 *
 * @return object Objet DB (base de données)
 */
function CC_connexion()
{
    global $typeServeur, $serveur, $base, $utilisateur, $motDePasse;

    switch ($typeServeur) {
        case "IBMDB2" :
        case "MSAccess" :
            $typeServeurPear = "odbc";
            break;
        default :
            $typeServeurPear = $typeServeur;
    }

    $dsn = "$typeServeurPear://";
    if (!empty($utilisateur)) $dsn .= $utilisateur;
    if (!empty($motDePasse))  $dsn .= ":$motDePasse";
    $dsn .= "@tcp($serveur)";
    if (!empty($base)) $dsn .= "/$base";

    $bd = DB::connect($dsn, TRUE);
    if (DB::isError($bd)) {
            echo $bd->message;
            return FALSE;
    }

    return $bd;
}

/**
 * Fonction de deconnexion.
 * (Ne fait rien sachant que la fonction
 * de connexion établit une connexion persistante)
 */
function CC_deconnexion($bd)
{
    return TRUE;
}

/**
 * Fonction chargé de créer et d'alimenter
 * la table "compteur de clics"
 **/
function CC_initialiseBD($bd, $table)
{
```

```
    global $typeServeur;

    $default  = "DEFAULT 0";

    // Création de la table
    $requete = "DROP TABLE $table";
    @$bd->query($requete);

    $requete = "CREATE TABLE $table (".
                              "urlId INTEGER PRIMARY KEY,".
                              "url VARCHAR(128) NOT NULL,".
                              "nbclic INTEGER $default,".
                              "UNIQUE(url))";
    $sts = $bd->query($requete);
    if (DB::isError($sts)) {
            echo $sts->message;
            return FALSE;
    }

    $bd->dropSequence($table);

    $sts = $bd->createSequence($table);
    if (DB::isError($sts)) {
            echo $sts->message;
            return FALSE;
    }

    // Alimentation de la table
    $requete     = "INSERT INTO $table (urlId,url) VALUES (?,?)";
    $reqPreparee = $bd->prepare($requete);

    $tabUrl = array (
                array ($bd->nextId($table), 'http://www.php.net'),
                array ($bd->nextId($table), 'http://www.phpfacile.com'),
                array ($bd->nextId($table), 'http://www.sqlfacile.com'),
                array ($bd->nextId($table), 'http://www.xmlfacile.com'),
                array ($bd->nextId($table), 'http://www.ootoogo.com')
                  );
    $sts = $bd->executeMultiple($reqPreparee, $tabUrl);
    if (DB::isError($sts)) {
            echo $sts->message;
            return FALSE;
    }

    return TRUE;
}

/**
 * Fonction retournant les informations de liens
 * sous forme d'un tableau associatif possedants
 * les clés
```

```
     * - "lien" associé au tableau des liens hypertextes
     * - "nbclic" associé au tableau des nombres de clics
     **/
    function CC_recupereLiens($bd, $table)
    {
        // Nom du script chargé du comptage et de la redirection
        $script = "compteurclic_redirection.php";

        // Requête SELECT
        $requete = "SELECT * FROM $table";
        $resultat = $bd->query($requete);
        if (DB::isError($resultat)) return FALSE;

        // Récupération des enregistrements les uns après les autres
        while ($enreg = $resultat->fetchRow(DB_FETCHMODE_ASSOC)) {
            $liens["lien"][] = "<a href=\"$script?urlid=".
                               $enreg["urlId"]."\">".
                               $enreg["url"]."</a>";
            $liens["nbclic"][] = $enreg["nbclic"];
        }

        return $liens;
    }

    /**
     * Fonction récupérant une url à partir de son identifiant
     * et incrémentant le compteur de clics
     **/
    function CC_recupereUrl($bd, $urlid)
    {
        global $table;

        // Récupère l'url
        $requete = "SELECT * FROM $table WHERE urlid=$urlid";
        $enreg = $bd->getRow($requete, NULL, DB_FETCHMODE_ASSOC);
        if (DB::isError($enreg)) return FALSE;

        if ($enreg) {
            $url = $enreg["url"];

            // Incrémente le compteur
            $requete = "UPDATE $table SET nbclic=nbclic+1 WHERE urlid=$urlid";
            $bd->query($requete);
        } else {
            $url = FALSE;
        }
        return $url;
    }
?>
```

Grâce à l'utilisation des séquences PEAR, ce script pourrait s'adapter à différents types de bases de données, puisqu'il s'affranchit des disparités existantes dans la déclaration des champs auto-incrémentés. Mais, malheureusement, cela n'est pas suffisant, puisqu'il devrait être également adapté, par exemple, pour MS Access, qui ne supporte pas l'instruction SQL "DEFAULT". Il devrait de même être adapté pour MS SQL Server, puisque nous avons constaté un dysfonctionnement lié à l'utilisation des méthodes `prepare()`, `execute()` ou `executeMultiple()`.

La couche d'abstraction PEAR ne permet donc pas de pallier tous les problèmes de compatibilité.

Les scripts contenant les fonctions d'affichage, quant à eux, ne sont pas modifiés (d'où l'intérêt de les dissocier des fonctions de traitement). D'ailleurs, les scripts principaux ne le sont pas non plus, puisqu'ils n'accèdent pas directement à la base de données, n'appellant que des fonctions de "haut niveau".

SuperTheque

L'essentiel du code de l'application a été présenté en introduction de ce chapitre. Il ne restait plus qu'à connaître les fonctions PEAR DB pour implémenter l'interface `RessourceInterface` c'est désormais chose faite:

Listing 10.108 : RessourcePEARDB_class.php

```
<?php
include_once("RessourceInterface_class.php");
include_once(dirname(__FILE__)."/../config/peardb_cfg.php");
include_once("DB.php");
/**
 * RessourcePEARDB_class.php
 * Classe d'acces a une base de donnees via PEARDB
 * Cette classe doit implementer toutes les methodes de l'interface
 *   RessourceInterface.
 * Compatibilite: PHP 5
 */
class Ressource implements RessourceInterface
{
    var $bd;

    public function connexion()
    {
        global $pear_typeServeur;
        global $pear_serveur, $pear_utilisateur, $pear_motDePasse;
        global $pear_base;

        switch ($pear_typeServeur) {
            case "IBMDB2" :
            case "MSAccess" :
                $typeServeurPear = "odbc";
                break;
            default :
                $typeServeurPear = $pear_typeServeur;
```

```
    }

    $dsn = "$typeServeurPear://";
    if (!empty($pear_utilisateur)) $dsn .= $pear_utilisateur;
    if (!empty($pear_motDePasse))  $dsn .= ":$pear_motDePasse";
    $dsn .= "@tcp($pear_serveur)";
    if (!empty($pear_base)) $dsn .= "/$pear_base";

    $bd = DB::connect($dsn, TRUE);
    if (DB::isError($bd)) {
            echo $dsn." ".$bd->message;
            return FALSE;
    }

    $this->bd = $bd;
    return TRUE;
}

public function deconnexion()
{
    // Rien a faire, il s'agit d'une connexion persistante
}

public function reset()
{
    $requete = "DROP TABLE types";
    $resultat = $this->bd->query($requete);
    // Pas de raison de retourner un erreur
    // si la suppression echoue (si la table n'existe pas)

    $requete = "CREATE TABLE types (".
                    "id INTEGER PRIMARY KEY,".
                    "type VARCHAR(255))";
    $resultat = $this->bd->query($requete);
    if (DB::isError($resultat)) {
            echo $resultat->message;
            return FALSE;
    }

    $this->bd->dropSequence("types");

    $resultat = $this->bd->createSequence("types");
    if (DB::isError($resultat)) {
            echo $resultat->message;
            return FALSE;
    }


    $types = array("Album", "Film", "Livre", "Musique");
    foreach ($types as $type) {
        $requete = "INSERT INTO types (id, type) VALUES (".
                   $this->bd->nextId("types").",'$type')";
```

```
        $resultat = $this->bd->query($requete);
        if (DB::isError($resultat)) {
            echo $resultat->message;
            return FALSE;
        }
    }

    $requete = "DROP TABLE articles";
    $resultat = $this->bd->query($requete);
    // Pas de raison de retourner un erreur
    // si la suppression echoue (si la table n'existe pas)

    $requete = "CREATE TABLE articles (".
                "id INTEGER PRIMARY KEY,".
                "albumId INTEGER,".
                "titre VARCHAR(255),".
                "typeId INTEGER,".
                "commentaire VARCHAR(255))";
    $resultat = $this->bd->query($requete);
    if (DB::isError($resultat)) {
            echo $resultat->message;
            return FALSE;
    }

    $this->bd->dropSequence("articles");

    $resultat = $this->bd->createSequence("articles");
    if (DB::isError($resultat)) {
            echo $resultat->message;
            return FALSE;
    }

}

public function addArticle($albumId,
                           $titre,
                           $typeId,
                           $commentaire)
{
    $requeteDebut = "INSERT INTO articles (id, albumId, titre, typeId";
    $requeteFin = ") VALUES (".$this->bd->nextId("articles").",".
                             "$albumId,".
                             "'".$this->bd->escapeSimple($titre)."',".
                             "$typeId";
    if ($commentaire != "") {
         $requeteDebut .= ",commentaire";
         $requeteFin .= ",'".$this->bd->escapeSimple($commentaire)."'";
    }
    $requete = $requeteDebut . $requeteFin . ")";
    $resultat = $this->bd->query($requete);
    if (DB::isError($resultat)) {
            echo $resultat->message;
```

```
            return FALSE;
        }
    }

    public function getArticle($articleId)
    {
        $requete = "SELECT * FROM articles WHERE id=$articleId";
        $resultat = $this->bd->query($requete);
        if (DB::isError($resultat)) {
                echo $resultat->message;
                return FALSE;
        }

        $enreg = $this->bd->getRow($idResultat);
        if (DB::isError($enreg)) return NULL;
        $article = $this->enreg2Article($enreg);
        return $article;
    }

    public function getArticles(Filtre $filtre, Plage $plage, Tri $tri)
    {
        $requete = "SELECT * FROM articles";
        if ($filtre->getAlbumId() != -1) {
            $conditionSQL = "albumId=".$filtre->getAlbumId();
        }
        if ($filtre->getTitre() != "") {
            $conditionSQL = $conditionSQL.
                            " AND titre LIKE '".
                            $this->bd->escapeSimple($filtre->getTitre())."'";
        }
        if ($filtre->getTypeId() != -1) {
            $conditionSQL = $conditionSQL.
                            " AND typeId=".$filtre->getTypeId();
        }

        if ($tri->getSens() == -1) {
            $triSQL = "ORDER BY ".$tri->getChamp()." DESC";
        } else if ($tri->getSens() == 1) {
            $triSQL = "ORDER BY ".$tri->getChamp()." ASC";
        }

        $requete = $requete." WHERE ".$conditionSQL." ".
                                      $triSQL;
        $resultat = $this->bd->limitQuery($requete,
                                          $plage->getPremierArticle(),
                                          $plage->getNbArticleMax());
        if (DB::isError($resultat)) {
                echo $resultat->message;
                return FALSE;
        }

        $articles = NULL;
```

```
    while ($enreg = $resultat->fetchRow(DB_FETCHMODE_ASSOC)) {
        $articles[] = $this->enreg2Article($enreg);
    }
    return $articles;
}

public function getNbTotalArticles(Filtre $filtre)
{
    $requete = "SELECT COUNT(*) FROM articles";
    if ($filtre->getAlbumId() != -1) {
        $conditionSQL = "albumId=".$filtre->getAlbumId();
    }
    if ($filtre->getTitre() != "") {
        $conditionSQL = $conditionSQL.
                        " AND titre LIKE '".
                        $this->bd->escapeSimple($filtre->getTitre())."'";
    }
    if ($filtre->getTypeId() != -1) {
        $conditionSQL = $conditionSQL.
                        " AND typeId=".$filtre->getTypeId();
    }

    $requete = $requete." WHERE ".$conditionSQL;
    $resultat = $this->bd->getOne($requete);
    if (DB::isError($resultat)) {
            echo $resultat->message;
            return FALSE;
    }
    return $resultat;
}

public function getTypes()
{
    $requete = "SELECT * FROM types";
    $resultat = $this->bd->query($requete);
    if (DB::isError($resultat)) {
            echo $resultat->message;
            return FALSE;
    }

    $types = NULL;
    while ($enreg = $resultat->fetchRow(DB_FETCHMODE_ASSOC)) {
        $types[$enreg["id"]] = $enreg["type"];
    }
    return $types;
}

public function getAlbumTypeId()
{
     return 1;
}
```

```
    private function enreg2Article($enreg)
    {
        $article = new Article();
        $article->setId($enreg["id"]);
        $article->setAlbumId($enreg["albumId"]);
        $article->setTitre($enreg["titre"]);
        $article->setTypeId($enreg["typeId"]);
        $article->setCommentaire($enreg["commentaire"]);
        return $article;
    }

}
```

En revanche, l'utilisation des méthodes `PEAR` permettant la manipulation des séquences, si elle n'est pas optimale, permet toutefois de s'affranchir des différences entre les serveurs de bases de données concernant les champs auto-incrémentés.

En savoir plus...

Il est possible de récupérer un certain nombre d'informations sur le serveur de bases de données (ses bases, ses tables, etc.) grâce à la méthode `getListOf()`.

DB->getListOf()

Retourne la liste des bases, tables, etc.

Syntaxe	`array getListOf(string $cle)`
`$cle`	Mot-clé parmi : "`databases`" pour récupérer un tableau indexé des bases disponibles sur le serveur. "`tables`" pour récupérer un tableau indexé des tables disponibles dans la base à laquelle vous êtes connecté. "`users`" pour récupérer la liste des utilisateurs. "`views`" pour récupérer la liste des vues disponibles dans la base à laquelle vous êtes connecté. "`functions`" pour récupérer la liste des fonctions.
`retour`	Le tableau indexé demandé, ou *DB_Error* en cas d'échec.

Il est également possible de tester les fonctionnalités offertes par le serveur de bases de données ou la façon dont `PEAR` les gère.

DB->provides()

Teste la gestion d'une fonctionnalité par PEAR ou le serveur de bases de données.

Syntaxe	`string provides(string $fonctionnalite)`
`$fonctionnalite`	Fonctionnalité à tester (ex. : "`prepare`", "`pconnect`", "`transactions`", "`limit`", etc.)
`retour`	Selon les cas : `TRUE` si la fonctionnalité est supportée par le serveur de bases de données. "`alter`" si PEAR a besoin de modifier la requête de l'utilisateur pour l'utiliser. "`emulate`" si PEAR émule la fonctionnalité. `FALSE` si elle n'est pas supportée par le serveur. `DB_error` en cas d'échec.

Chapitre 11

Les annuaires LDAP

Aux origines du standard LDAP, on retrouve des organisations internationales de normalisation. En effet, de grands groupes de télécommunications ainsi que l'International Organization for Standardization (ISO), avaient la volonté de normaliser le protocole de messagerie. Après la publication de leurs travaux (la norme X400), ils développèrent la norme censée interconnecter les annuaires des différents pays. C'est ainsi que naquirent les normes X500.

Protocole LDAP

LDAP (Lightweight Directory Access Protocol) est un protocole permettant la standardisation de l'accès à des données provenant d'un annuaire, permettant ainsi à différentes applications d'accéder à une source commune.

Avec cette norme, il était enfin possible d'interroger, à partir de n'importe quelle application, tous les annuaires mondiaux ; ce qui pouvait donc, à terme, permettre de créer un vaste répertoire mondial.

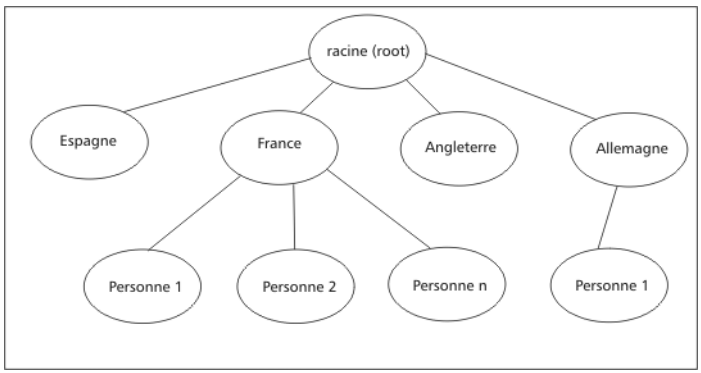


Figure 11.1 : *Principe d'un annuaire X500*

Une application cliente peut interroger l'annuaire à l'aide du protocole DAP (Directory Access Protocol). L'application passe alors par une couche d'abstraction : le DUA (Directory User Agent). Celui-ci interroge ensuite l'annuaire et retourne les informations au DUA, qui effectue alors le travail inverse en retournant les données au client. Ainsi, toutes les applications voulant profiter d'un annuaire central devaient simplement développer la couche d'abstraction à l'aide d'une API (voir fig. 11.2).

L'un des intérêts du protocole X500 est également la possibilité de placer différents serveurs d'annuaires en parallèle, les annuaires dialoguant entre eux à l'aide du protocole DSP (Directory System Protocol).

Les normes X500 ont également standardisé le schéma que peut avoir un annuaire. Les différentes données sont organisées suivant leur appartenance à des classes d'objets, qui elles-mêmes sont définies par différents attributs. Ces attributs peuvent posséder une ou plusieurs valeurs. Chaque classe définie par rapport à une classe parente hérite de tous les attributs de la classe mère dont elle dérive. Afin d'éviter certains problèmes, la norme a également adopté la standardisation des annuaires décrivant les personnes, les organisations, etc.

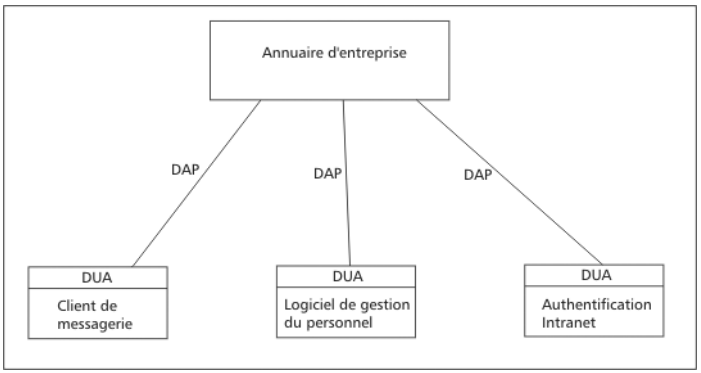


Figure 11.2 : *Exemple d'application d'un annuaire au sein d'une entreprise*

La complexité de la mise en œuvre d'un schéma complet rend son utilisation complexe pour une petite structure de type PMI ou PME. De plus, le DAP permettant la connexion des clients se trouve être très complexe à utiliser, car il nécessite l'équipement du poste client en matériel spécifique. En effet, le protocole n'utilise pas TCP/IP, et, de ce fait, il est nécessaire d'intégrer sur les clients une carte réseau de type X25. L'arrivée en force d'Internet pousse donc un nouveau groupe à se pencher sur la problématique. Ainsi, en se basant sur les normes X500 et le support du protocole TCP/IP, est publiée la version 2 de LDAP en 1994, puis la version 3 en 1997. Vous pouvez retrouver les spécifications (en anglais) de cette dernière version dans les RFC qui vont de 2251 à 2256 aux adresses allant de http://www.ietf.org/rfc/rfc2251.txt à http://www.ietf.org/rfc/rfc2256.txt. (des traductions en français sont disponibles au format PDF depuis des liens proposés sur http://abcdrfc.free.fr).

Les avantages de cette norme sont :

- Support du modèle de données X500 ;
- Ouverture sur l'internet par son support TCP/IP ;
- Sécurisation de l'accès aux données par personne ou groupe de personnes ;
- Les requêtes sont normalisées permettant ainsi une recherche plus efficace ;
- LDAP permet l'interrogation de plusieurs annuaires de façon transparente pour le client.

Il existe plusieurs annuaires. Des annuaires commerciaux (sites en anglais) :

- eDirectory de Novell (http://www.novell.com/products/edirectory/) ;
- Active Directory de Microsoft (http://www.microsoft.com/windows2000/technologies/directory/AD/default.asp) ;
- SunOne Directory Server de Sun (http://wwws.sun.com/software/products/directory_srvr/home_directory.html) ;
- Oracle (http://www.oracle.com) ;
- IBM Directory Server (http://www-3.ibm.com/software/network/directory/).

Ainsi que quelques (trop) rares annuaires libres (sites en anglais) :

- OpenLDAP (http://www.openldap.org) ;
- Apple Open Directory (http://developer.apple.com/darwin/projects/opendirectory/).

11.1. Le schéma LDAP

Le principe de LDAP est expliqué à partir de ce schéma :

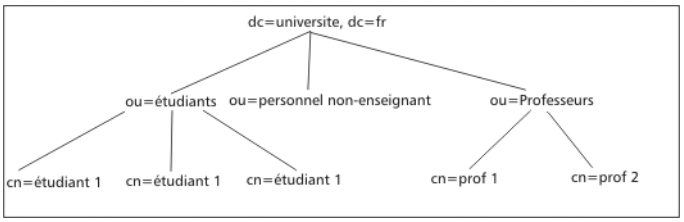


Figure 11.3 : *Vision classique d'un annuaire LDAP*

Afin de nommer chaque classe, on utilise un ou plusieurs attributs les définissant. Ainsi, on commence par la classe `racine` qui possède un DN (Distinguished Name) égal à `dc=universite, dc=fr`. Suivent les classes représentées par leur RDN (Relative Distinguished Name), le DN unique d'une entrée étant l'ajout de son RDN au DN de la classe mère de l'objet séparé par une virgule. Ainsi, l'entrée représentée par le RDN égal à `ou=étudiants` possède un DN équivalent à `ou=étudiants, dc=universite, dc=fr`.

Ainsi, le DN de l'étudiant 1 est : `cn=étudiant 1, ou=étudiants, dc=universite, dc=fr`.

Le DN du professeur 2 est : `cn=prof 2, ou=Professeurs, dc=universite, dc=fr`.

Nous n'avons pas la prétention dans cet ouvrage de vous faire connaître sur le bout des doigts le protocole LDAP et son utilisation. Internet regorge de liens qui devraient pouvoir vous être utiles dans votre apprentissage des annuaires. Vous pouvez toujours commencer par visiter le site web du CRU (Comité Réseau des Universités), qui possède une page très intéressante avec de nombreux liens et de documents : http://www.cru.fr/ldap/.

11.2. Installation

Nous nous contenterons ici de décrire une installation standard sans tenir compte des problèmes d'optimisation et de sécurité. Le but est que vous puissiez installer PHP et LDAP sur des machines de test, pour vous familiariser avec cet environnement avant de passer à un serveur d'annuaire destiné à la production. Pour des raisons de coût évidentes, nous avons opté pour un serveur OpenLDAP.

Installation du serveur OpenLDAP

L'installation du serveur LDAP est facilitée par sa mise à disposition sur certaines distributions Linux. Malgré tout, pour ceux qui désirent compiler le serveur pour leur plateforme (par

exemple pour ceux qui utilisent un environnement Windows avec Cygwin installé), nous allons rapidement décrire son installation standard.

Installation

Vous devez, avant toute chose, récupérer l'archive qui se trouve sur le site d'OpenLDAP à l'adresse **http://www.openldap.org/software/download/** (nous mettons à votre disposition la version 2.1.3 sur le CD-ROM d'accompagnement de ce livre).

Dans un premier temps, décompactez l'archive à l'aide de la commande suivante :

```
$ tar zxvf openldap-2.1.3.tgz
$ cd openldap-2.1.3
```

Passez ensuite à la configuration, puis à la compilation.

```
$ ./configure
$ make depend
$ make
```

Passez en mode administrateur en tapant la commande `su` – et en entrant votre mot de passe `root`.

```
# make install
```

Configuration

Ouvrez ensuite votre éditeur fétiche, et éditez le fichier */usr/local/etc/openldap/slapd.conf*. Modifiez les paramètres `suffix`, `rootdn` et `rootpw` de la façon suivante :

```
# Indiquez ici la racine correspondant à votre domaine
suffix "dc=ldap, dc=tild, dc=com"
# Indiquez ici le DN de l'administrateur de l'annuaire
rootdn "cn=root, dc=ldap, dc=tild, dc=com"
# Indiquez ensuite le mot de passe de l'administrateur
rootpw coucou
```

Dans notre cas, le domaine spécifié est `ldap.tild.com` (ces paramètres doivent indiquer le nom de votre domaine).

Vérifiez que le répertoire spécifié par le paramètre `directory` existe bien.

Lancez le serveur LDAP avec la commande :

```
/usr/local/libexec/slapd
```

Vous pouvez vérifier que le serveur est bien lancé en exécutant la commande suivante :

```
ldapsearch -xs base '(objectclass=*)' namingContexts
```

La commande doit vous retourner un résultat similaire à celui-ci :

```
#
```

```
# filter: (objectclass=*)
# requesting: namingContexts
#

# ldap,dc=tild,dc=com
dn: dc=ldap,dc=tild,dc=com

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```


Il faut ensuite créer le schéma racine de l'annuaire. Pour cela, vous allez créer un fichier texte au format *LDIF*. Ouvrez votre éditeur (toujours le même, ou changez si vous voulez…) et entrez les lignes suivantes :

```
dn: dc=ldap,dc=tild,dc=com
objectclass: dcObject
objectclass: organization
o: Tild
dc: tild

dn: cn=root,dc=ldap,dc=tild,dc=com
objectclass: organizationalRole
cn: root
```

Enregistrez-le sous *monpremierschema.ldif* par exemple, et appelez la commande suivante :

```
ldapadd -x -D " cn=root,dc=ldap,dc=tild,dc=com" -W -f monpremierschema.ldif
```

Voilà, votre annuaire est prêt à recevoir de nouvelles entrées. Maintenant, nous n'allons pas continuer à les ajouter avec le client `ldapadd` : préférons (c'est un peu le sujet de ce livre) utiliser le langage PHP. Mais avant toute chose, installons le module LDAP.

Installation du module LDAP pour PHP

Installation sous Windows

Avec l'archive du PHP Group

Vous devez, dans un premier temps, vous assurer de posséder un fichier *php_ldap.dll* (fourni avec l'archive du PHP Group) dans le répertoire contenant vos extensions PHP. Il vous suffit ensuite de modifier le fichier *php.ini* pour ajouter ou décommenter une ligne.

```
extension=php_ldap.dll
```

Avec EasyPHP

L'installation d'EasyPHP possède, par défaut, le module de connexion à LDAP.

Installation sous Linux

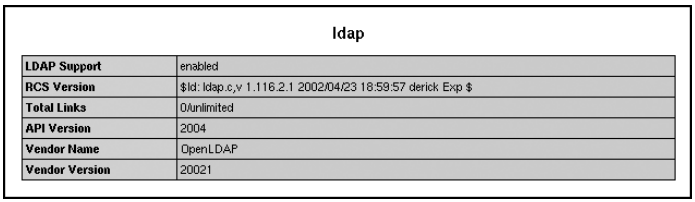
Pour installer le module LDAP, vous devez posséder les librairies installées sur votre machine. Si vous avez suivi l'installation du serveur LDAP de notre première partie, vous les possédez certainement.

Ensuite, recompilez PHP avec l'option suivante : *--with-ldap*.

Vous pouvez vérifier la bonne installation du module en créant une page qui ne contient que le code suivant :

```
<?php
    phpinfo();
?>
```

Lancez un navigateur et appelez la page. Vous devez apercevoir les lignes suivantes :



ldap

LDAP Support	enabled
RCS Version	$Id: ldap.c,v 1.116.2.1 2002/04/23 18:59:57 derick Exp $
Total Links	0/unlimited
API Version	2004
Vendor Name	OpenLDAP
Vendor Version	20021

Figure 11.4 : *La fonction phpinfo() vous renseigne sur les modules qui sont installés. Vérifiez que la section LDAP existe.*

11.3. L'interrogation de LDAP avec PHP

L'utilisation simple de PHP pour interroger un annuaire LDAP s'opère selon le schéma suivant :

- Connexion à un serveur LDAP ;
- Identification sur serveur ;
- Opérations sur la base LDAP ;
- Fermeture de la connexion avec le serveur LDAP.

Connexion, authentification et déconnexion sur le serveur LDAP

Connexion sur l'annuaire

Pour vous connecter au serveur LDAP, vous devrez utiliser la fonction `ldap_connect()`.

ldap_connect()

Établit une connexion entre le client (PHP) et l'annuaire LDAP.

Syntaxe	`resource ldap_connect([string $annuaire [, int $port]])`
`$serveur`	Adresse de l'annuaire LDAP. Par défaut, la connexion s'effectue sur le serveur local.
`$port`	Port sur lequel on doit se connecter. Par défaut, le port est le 389.
`retour`	Identifiant de la connexion à l'annuaire.

Gérer l'erreur de connexion
Lorsque le serveur n'existe pas ou que la connexion n'a pas été effectuée, l'instruction `ldap_connect()` *ne retourne pas d'erreur. Pour gérer les erreurs de liaison avec l'annuaire, vous devez, au préalable, exécuter la fonction* `ldap_bind()`.

Authentification sur un annuaire

L'instruction `ldap_bind()` vous permettra de vous identifier auprès du serveur LDAP.

ldap_bind()

Effectue une liaison entre l'annuaire et le client LDAP (PHP dans notre cas).

Syntaxe	`boolean ldap_bind(resource $idLdap [, string $dnUtilisateur [, string $motDePasse]])`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`$dnUtilisateur`	DN (Distinguished Name) de l'utilisateur qui demande la connexion. Ce paramètre est optionnel ; si vous ne le précisez pas, la connexion s'effectuera en mode anonyme.
`$motDePasse`	Mot de passe de l'utilisateur demandant la connexion. Ce paramètre est optionnel.
`retour`	`TRUE` si la liaison avec l'annuaire a été effectuée, `FALSE` dans le cas contraire.

Déconnexion de l'annuaire

La déconnexion de l'annuaire s'effectue avec la fonction `ldap_unbind()` ou son alias `ldap_close()`.

ldap_close()

Se déconnecte du serveur LDAP.

Syntaxe	`boolean ldap_unbind(resource $idLdap)`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`retour`	`TRUE` si la connexion a bien été fermée, `FALSE` dans le cas contraire.

Changer le DN de la connexion LDAP

Ne confondez pas `ldap_unbind()` avec une fonction qui pourrait permettre de changer l'utilisateur connecté à l'annuaire. Malgré son nom, l'instruction `ldap_unbind()` ferme la connexion, et vous devrez alors exécuter une nouvelle fois la fonction `ldap_connect()`. Pour changer d'utilisateur, appelez simplement de nouveau `ldap_bind()` avec un nouveau DN.

Opérations sur un annuaire LDAP

Vous pouvez effectuer quatre opérations standard sur les entrées d'un serveur LDAP :

- Ajouter une entrée ;
- Modifier une entrée ;
- Supprimer une entrée ;
- Renommer une entrée.

Ajouter une entrée

L'ajout d'une entrée dans l'annuaire s'effectue avec l'instruction `ldap_add()`.

ldap_add()

Ajoute une entrée dans un annuaire.

Syntaxe	`boolean ldap_add(resource $idLdap, string $dn, array $attributs)`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`$dn`	DN de l'entrée à ajouter.
`$attributs`	Tableau associatif contenant les différents attributs de l'objet à ajouter.
`retour`	`TRUE` si l'objet a bien été ajouté, `FALSE` dans le cas contraire.

Pour ajouter une entrée, vous devez créer un tableau associatif contenant tous les attributs à ajouter en clés ainsi que les valeurs de ces attributs. Si un attribut peut contenir plusieurs valeurs, celles-ci sont alors rentrées dans un nouveau tableau.

```
<?php
$attributs['attribut1'] = valeur1;
$attributs['attribut2'] = valeur2;
// Attribut contenant plusieurs valeurs
$attributs['attribut3'][0] = valeur3;
$attributs['attribut3'][1] = valeur4;
?>
```

Voici un exemple d'ajout dans l'annuaire avec un script PHP.

Listing 11.1 : ldap_add.php

```
<?php
// Connexion à l'annuaire
$idLdap = ldap_connect("localhost",389);

$rootDn    = 'cn=root,dc=ldap,dc=tild,dc=com';
$rootPasse = 'coucou';

if (!ldap_bind($idLdap, $rootDn, $rootPasse)){
    die("La connexion n'a pas été effectuée.");
}

// On construit le tableau avec les différents attributs
$attributs['o']           = 'Tild';
$attributs['givenname']   = 'Laurent';
$attributs['street']      = '5 rue Nominoe';
$attributs['sn']          = 'GUEDON';
$attributs['l']           = 'RENNES';
$attributs['objectclass'] = 'person';
$attributs['mail'][0]     = 'tendencies@free.fr';
$attributs['mail'][1]     = 'laurent@tild.com';
$attributs['cn']          = 'lolo';

// DN de l'entrée à ajouter
$dn = 'cn=lolo,dc=ldap,dc=tild,dc=com';

if (@ldap_add($idLdap, $dn, $attributs))
{
    echo "L'entrée a été ajoutée !";
}else{
    echo "Problème, l'entrée n'a pas été ajoutée !";
}

// Déconnexion de l'annuaire
ldap_close($idLdap);
?>
```

Modifier une entrée

Modifier les attributs d'une entrée

Pour la modification des entrées, nous utiliserons la fonction `ldap_mod_replace()` ou son alias `ldap_modify()`.

ldap_mod_replace()

Modifie une entrée de l'annuaire LDAP. Cette fonction peut affecter plusieurs attributs à la fois.

Syntaxe	`boolean ldap_mod_replace(resource $idLdap, string $dn, array $attributs)`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`$dn`	DN de l'entrée à modifier.
`$attributs`	Tableau associatif contenant les différents attributs de l'objet qui doivent être modifiés.
`retour`	`TRUE` si l'entrée a bien été modifiée dans l'annuaire, `FALSE` dans le cas contraire.

La création du tableau `$entree` est identique à celle de `ldap_add()`. Vous devez spécifier dans le tableau les différents attributs qui doivent être modifiés.


REMARQUE

Modification d'un attribut à valeur multiple

Attention aux attributs contenant plusieurs valeurs. Si vous ne modifiez qu'une seule de ces valeurs, alors l'attribut supprimera les autres en conservant la valeur modifiée. Prenons un exemple et imaginons l'entrée multiple suivante :

```
<?php
$attributs['mail'][0] = "webmaster@tild.com";
$attributs['mail'][1] = "info@tild.com";
$attributs['mail'][2] = "contact@tild.com";
?>
```

Si vous pensez remplacer "contact@tild.com" par "laurent@tild.com", et donc que vous modifiez l'entrée de cette façon :

```
<?php
$attributs['mail'][2] = "laurent@tild.com";
ldap_mod_replace($idLdap, "cn=lolo,dc=ldap,dc=tild,dc=com", $attributs);
?>
```

vous supprimez alors les autres valeurs de l'attribut multiple `mail`, et ne retrouvez que la valeur "laurent@tild.com". Vous pouvez le vérifier en rentrant la commande sur votre console :

```
$ ldapsearch -x -b 'dc=ldap,dc=tild,dc=com' '(objectclass=*)'
```

Ajouter des attributs à une entrée

L'instruction `ldap_mod_add()` permet d'ajouter un attribut à une entrée déjà existante.

ldap_mod_add()

Ajoute un attribut à une entrée de l'annuaire LDAP.

Syntaxe	`boolean ldap_mod_add(resource $idLdap, string $dn, array $attributs)`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`$dn`	DN de l'entrée devant contenir les attributs à ajouter.
`$attributs`	Tableau associatif contenant les différents attributs à ajouter ainsi que leurs valeurs. Si un attribut est déjà existant, la fonction lui ajoute une nouvelle valeur. Il est possible de spécifier l'ajout d'un attribut à valeur multiple de la même façon que l'instruction `ldap_add()`.
`retour`	`TRUE` si les attributs ont bien été ajoutés, `FALSE` dans le cas contraire.

```
<?php
// Voici le tableau contenant les attributs à ajouter
$attributs['description'] = 'Un des auteurs de la bible du PHP';
// Si l'entrée existe, alors "tendencies@free.fr" sera
// ajouté comme une nouvelle valeur de l'attribut mail
$attributs['mail'] = 'tendencies@free.fr';
ldap_mod_add($idLdap, $baseDn, $attributs);
?>
```

Supprimer les attributs d'une entrée

Il est aussi possible de supprimer l'attribut d'une entrée existante. Pour cela, nous ferons appel à la fonction `ldap_mod_del()`.

ldap_mod_del()

Supprime un attribut d'une entrée de l'annuaire LDAP.

Syntaxe	`boolean ldap_mod_del(resource $idLdap, string $dn, array $attributs)`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.

`$dn`	DN de l'entrée contenant les attributs à supprimer.
`$attributs`	Tableau associatif contenant les différents attributs à supprimer. Les valeurs de chaque attribut doivent être indiquées. Cela permet de supprimer simplement une seule valeur lorsque l'attribut en possède plusieurs. Tous les attributs doivent exister ; dans le cas contraire, l'instruction retourne une erreur et n'en supprimera aucune.
`retour`	`TRUE` si la ou les attributs désignés ont bien été supprimés, `FALSE` dans le cas contraire.

```
// Voici le tableau contenant les attributs à supprimer
$entree['description'] = 'Un des auteurs de la bible PHP';
$entree['mail']        = 'tendencies@free.fr';
$dn = 'cn=lolo,dc=kangouroo,dc=homelinux,dc=org';

if (ldap_mod_del($idLdap, $dn, $entree))
{
    echo "Les attributs ont été supprimés !";
}else{
    echo "Problème, les attributs n'ont pas été supprimés !";
}
```


Supprimer une entrée de l'annuaire

La suppression d'une entrée est réalisée par l'appel à la fonction `ldap_delete()`.

ldap_delete()

Supprime une entrée d'un annuaire LDAP.

Syntaxe	`boolean ldap_delete(resource $idLdap, string $dn)`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`$dn`	Le DN (Distinguished Name) de l'entrée à supprimer.
`retour`	`TRUE` si l'opération de suppression a été exécutée avec succès, `FALSE` dans le cas contraire.

Vous ne pouvez pas supprimer une entrée qui n'est pas vide. Pour effectuer cette opération, vous devez, dans un premier temps, supprimer toutes les entrées existantes dans l'entrée père. Nous effectuerons cette opération à l'aide d'une fonction récursive.

Listing 11.2 : ldap_delete.inc.php

```
<?php
/**
 * Fonction de suppression des entrées récursives ou non
 * @input $dn string le DN de l'élément à supprimer
```

```
 *        $recursive boolean Indique si la fonction
 *          doit suppprimer récursivement
 * @return boolean reussite ou échec de l'opération
 */
function supprimerEntree($idLdap, $dn, $recursive=FALSE)
{
    if ($recursive){
        // On récupère les entrées fils
        $reponse = ldap_list($idLdap, $dn, "ObjectClass=*");
        $entree  = ldap_get_entries($idLdap, $reponse);
        for($i=0;$i<$entree['count'];$i++){
            // On supprime les entrées trouvées en appellant
            // la fonction supprimerEntree
            if (!supprimerEntree($idLdap, $entree[$i]['dn'], TRUE))
            {
                return FALSE;
            }
        }
    }
    // On supprime l'entrée courante
    if (ldap_delete($idLdap, $dn)) {
        return TRUE;
    } else {
        return FALSE;
    }
}
?>
```

À présent, venons-en à l'utilisation de cette fonction :

Listing 11.3 : ldap_delete.php

```
<?php
include("ldap_delete.inc.php ");

// Connexion à l'annuaire
$idLdap = ldap_connect("localhost",389);

$rootDn = 'cn=root,dc=ldap,dc=tild,dc=com';
$rootPasse = 'coucou';

if (!ldap_bind($idLdap, $rootDn, $rootPasse)){
    die("La connexion n'a pas été effectuée.");
}
// DN de l'entrée à créer et ensuite à supprimer
$baseDn = 'o=test,dc=ldap,dc=tild,dc=com';

// Création d'une entrée "organisation"
$groupe['objectclass'][0] = 'organization';
$groupe['objectclass'][1] = 'top';
$groupe['o']              = 'test '.$i;
ldap_add($idLdap, $baseDn, $groupe);
```

```
// création des membres dans le groupe
for ($i=0;$i<10;$i++)
{
    $entree['objectclass'][0] = 'top';
    $entree['objectclass'][1] = 'person';
    $entree['cn']             = 'personne'.$i;
    echo 'On ajoute l\'utilisateur personne'.$i.' dans l\'annuaire<br />';
    ldap_add($idLdap, 'cn=personne'.$i.",".$baseDn, $entree);
}

// Recherche et affichage des entrées dans le groupe
$reponse  = ldap_list($idLdap, $baseDn, "ObjectClass=*");
$resultat = ldap_get_entries($idLdap, $reponse);
echo "Il y a ".$resultat['count']." entrées dans le groupe<br />";
for($i=0;$i<$resultat['count'];$i++){
    echo "*".$resultat[$i]['dn']."<br />";
}

echo "Suppression de l'entrée récursivement<br />";
if (supprimerEntree($idLdap, $baseDn, TRUE))
{
    echo "L'entrée a été supprimée !<br />";
}else{
    echo "Problème, l'entrée n'a pas été supprimée !<br />";
}

// On vérifie le resultat de la suppression
$reponse = ldap_list($idLdap, "dc=ldap,dc=tild,dc=com",
                              "&((o=test)(ObjectClass=*))");
$resultat = ldap_get_entries($idLdap, $reponse);
if ($resultat['count']==0)
{
    echo "Il n'y a aucune entrée dn=o=test<br />";
}
// Déconnexion de l'annuaire
ldap_close($idLdap);
?>
```

Et voici maintenant le résultat de l'exécution de notre script :

```
On ajoute l'utilisateur personne0 dans l'annuaire
On ajoute l'utilisateur personne1 dans l'annuaire
On ajoute l'utilisateur personne2 dans l'annuaire
On ajoute l'utilisateur personne3 dans l'annuaire
On ajoute l'utilisateur personne4 dans l'annuaire
On ajoute l'utilisateur personne5 dans l'annuaire
On ajoute l'utilisateur personne6 dans l'annuaire
On ajoute l'utilisateur personne7 dans l'annuaire
On ajoute l'utilisateur personne8 dans l'annuaire
On ajoute l'utilisateur personne9 dans l'annuaire
Il y a 10 entrées dans le groupe
```

```
*cn=personne0,o=test,dc=ldap,dc=tild,dc=com
*cn=personne1,o=test,dc=ldap,dc=tild,dc=com
*cn=personne2,o=test,dc=ldap,dc=tild,dc=com
*cn=personne3,o=test,dc=ldap,dc=tild,dc=com
*cn=personne4,o=test,dc=ldap,dc=tild,dc=com
*cn=personne5,o=test,dc=ldap,dc=tild,dc=com
*cn=personne6,o=test,dc=ldap,dc=tild,dc=com
*cn=personne7,o=test,dc=ldap,dc=tild,dc=com
*cn=personne8,o=test,dc=ldap,dc=tild,dc=com
*cn=personne9,o=test,dc=ldap,dc=tild,dc=com
Suppression de l'entrée récursivement
L'entrée a été supprimée !
Il n'y a aucune entrée dn=o=test
```


Renommer une entrée

Pour renommer, copier ou déplacer une entrée, nous utiliserons la fonction `ldap_rename()`.

ldap_rename()

Renomme, copie ou déplace une entrée dans l'arbre LDAP.

Syntaxe	`boolean ldap_rename(int $idLdap, string $dn, string $nouveauRdn, string $nouvelleBaseDn, boolean $supprimer)`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`$dn`	Le DN (Distinguished Name) de l'entrée à supprimer.
`$nouveauRdn`	Nouveau RDN (Relative Distinguished Name) de l'entrée.
`$nouvelleBaseDn`	Nouvelle base DN parente pour l'entrée.
`$supprimer`	Indique s'il faut conserver l'ancienne entrée ou non.
`retour`	`TRUE` si l'entrée a bien été déplacée, `FALSE` dans le cas contraire.

La version du protocole v2 ne supporte pas cette fonction de PHP. Il faut nécessairement indiquer que nous utilisons la version 3 de LDAP pour utiliser `ldap_rename()`. Pour cela, utilisez l'instruction `ldap_set_option()` décrite plus loin.

Listing 11.4 : ldap_rename.php

```
<?php
$idLdap = ldap_connect("localhost",389);

// On passe en protocole v3 seul capable de supporter la copie
ldap_set_option($idLdap, LDAP_OPT_PROTOCOL_VERSION, 3);

// N'oubliez pas de modifier ces paramètres selon
```

```
// votre configuration
$rootDn    = 'cn=root, dc=tild';
$rootPasse = 'coucou';

if (!ldap_bind($idLdap, $rootDn, $rootPasse)){
    die("La connexion n'a pas été effectuée.");
}
// DN de l'entrée à copier
$dn = 'o=test,ou=test,dc=tild';
// nouveau DN
$rdn = 'o=test2';
// nouvelle base
$baseDn = 'ou=test,dc=tild';

// On déplace l'entrée.
$resultat = ldap_rename($idLdap, $dn, $rdn, $baseDn, TRUE);
if ($resultat)
{
    echo "L'entrée a été copiée !<br />";
} else {
    echo "L'entrée n'a pas été copiée !<br />";
    // Affichage de l'erreur
    echo ldap_error($idLdap);
}

// Déconnexion de l'annuaire
ldap_close($idLdap);
?>
```


Recherche dans un annuaire LDAP

La recherche dans un annuaire s'effectue en deux étapes :

- Lancement de la recherche ;
- Récupération des résultats.

Lancement d'une recherche dans un annuaire

La recherche dans un annuaire LDAP peut être effectuée de plusieurs façons :

- Rechercher une correspondance sur un niveau ;
- Rechercher une correspondance sur plusieurs niveaux ;
- Rechercher une correspondance sur une entrée spécifique.

Recherche dans un annuaire sur un niveau

La recherche sur un niveau s'effectue par l'appel à l'instruction `ldap_list()`.

ldap_list()

Recherche à l'aide d'un filtre les entrées sur un niveau.

Syntaxe	`resource ldap_list(resource $idLdap, string $baseDn, string $filtre [, array $attributs [, int $attributsSeul [, int $nbLimit [, int $tempsExecution [, int $alias]]]]])`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`$baseDn`	Le DN (Distinguished Name) indiquant la base de votre recherche.
`$filtre`	Filtre LDAP permettant de récupérer les informations demandées.
`$attributs`	Paramètre optionnel permettant de ne recevoir que les attributs désirés.
`$attributsSeul`	Paramètre indiquant de ne retourner que les noms des attributs sans leurs valeurs. `0` (valeur par défaut) indique de récupérer les valeurs et les attributs, `1` indique que seuls les attributs sont retournés.
`$nbLimit`	Indique le nombre de résultats à récupérer. Par défaut, la valeur est fixée à `0`, ainsi tous les résultats de la recherche sont renvoyés.
`$tempsExecution`	Indique la durée maximale d'une recherche en secondes. Par défaut, la valeur est fixée à `0`, c'est-à-dire qu'il n'y a pas de limite sinon celle fixée par le serveur LDAP.
`$alias`	Paramètre indiquant comment les alias des attributs doivent être considérés. Utilisez l'une des constantes suivantes : `LDAP_DEREF_NEVER` (valeur par défaut), indique que les alias ne peuvent être utilisés pour effectuer la recherche. `LDAP_DEREF_SEARCHING`, indique que les alias ne peuvent pas être utilisés pour la recherche, mais que, lors du renvoi des résultats, ceux-ci sont utilisés. `LDAP_DEREF_FINDING`, indique que les alias sont utilisés pour la recherche, mais que le résultat ne les prend pas en compte. `LDAP_DEREF_ALWAYS`, indique que les alias ne sont jamais pris en compte, ni pour la recherche ni pour le résultat.
`retour`	Pointeur sur le résultat de la recherche.

Recherche dans un annuaire sur plusieurs niveaux

Pour effectuer une recherche sur plusieurs niveaux, nous utiliserons la fonction `ldap_search()`.

ldap_search()

Recherche à l'aide d'un filtre les entrées sur plusieurs niveaux d'un annuaire LDAP. La recherche est effectuée sur le DN de la base, puis sur toutes les branches au-dessus.

Syntaxe	`resource ldap_search(resource $idLdap, string $baseDn, string $filtre [, array $attributs [, int $attributsSeul [, int $nbLimit [, int $tempsExecution [, int $alias]]]]])`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`$baseDn`	Le DN (Distinguished Name) indiquant le premier niveau où commencer votre recherche.
`$filtre`	Filtre LDAP permettant de récupérer les informations demandées.
`$attributs`	Paramètre optionnel permettant de ne recevoir que les attributs désirés.
`$attributsSeul`	Paramètre indiquant de ne retourner que les noms des attributs sans leurs valeurs. `0` (valeur par défaut) indique de récupérer les valeurs et les attributs, `1` indique que seuls les attributs sont retournés.
`$nbLimit`	Indique le nombre de résultats à récupérer. Par défaut, la taille est fixée à `0`, ainsi tous les résultats de la recherche sont renvoyés.
`$tempsExecution`	Indique la durée maximale d'une recherche en secondes. Par défaut, la valeur est fixée à `0`, c'est-à-dire qu'il n'y a pas de limite sinon celle fixée par le serveur LDAP.
`$alias`	Paramètre indiquant comment les alias des attributs doivent être considérés. Utilisez l'une des constantes suivantes : `LDAP_DEREF_NEVER` (valeur par défaut), indique que les alias ne peuvent être utilisés pour effectuer la recherche. `LDAP_DEREF_SEARCHING`, indique que les alias ne peuvent pas être utilisés pour la recherche, mais que, lors du renvoi des résultats, ceux-ci sont utilisés. `LDAP_DEREF_FINDING`, indique que les alias sont utilisés pour la recherche, mais que le résultat ne les prend pas en compte. `LDAP_DEREF_ALWAYS`, indique que les alias ne sont jamais pris en compte, ni pour la recherche ni pour le résultat.
`retour`	Pointeur sur le résultat de la recherche.

Recherche dans une entrée

Pour effectuer votre recherche sur une entrée et uniquement sur celle-ci, vous devez utiliser la fonction `ldap_read()`.

ldap_read()

Recherche à l'aide d'un filtre sur une entrée spécifique. La recherche est effectuée sur le DN spécifié en paramètre.

Syntaxe	`resource ldap_read(resource $idLdap, string $baseDn, string $filtre [, array $attributs [, int $attributsSeul [, int $nbLimit [, int $tempsExecution [, int $alias]]]]])`

`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`$baseDn`	Le DN (Distinguished Name) indiquant l'entrée où vous voulez effectuer votre recherche.
`$filtre`	Filtre LDAP permettant de récupérer les informations demandées.
`$attributs`	Paramètre optionnel permettant de ne recevoir que les attributs désirés.
`$attributsSeul`	Paramètre indiquant de ne retourner que les noms des attributs sans leurs valeurs. `0` (valeur par défaut) indique de récupérer les valeurs et les attributs, `1` indique que seuls les attributs sont retournés.
`$nbLimit`	Indique le nombre de résultats à récupérer. Par défaut, la taille est fixée à `0`, ainsi tous les résultats de la recherche sont renvoyés.
`$tempsExecution`	Indique la durée maximale d'une recherche en secondes. Par défaut la valeur est fixée à `0`, c'est-à-dire qu'il n'y a pas de limite sinon celle fixée par le serveur LDAP.
`$alias`	Paramètre indiquant comment les alias des attributs doivent être considérés. Utilisez l'une des constantes suivantes : `LDAP_DEREF_NEVER` (valeur par défaut), indique que les alias ne peuvent être utilisés pour effectuer la recherche. `LDAP_DEREF_SEARCHING`, indique que les alias ne peuvent pas être utilisés pour la recherche, mais que, lors du renvoi des résultats, ceux-ci sont utilisés. `LDAP_DEREF_FINDING`, indique que les alias sont utilisés pour la recherche, mais que le résultat ne les prend pas en compte. `LDAP_DEREF_ALWAYS`, indique que les alias ne sont jamais pris en compte, ni pour la recherche ni pour le résultat.
`retour`	Pointeur sur le résultat de la recherche.

Exemple de recherche

Cet exemple va vous montrer les différences existant entre ces trois fonctions :

Listing 11.5 : recherche.php

```
<html>
<body>

<?php
// Connexion à l'annuaire
$idLdap = ldap_connect("localhost",389);
if (!ldap_bind($idLdap)){
    die("La connexion n'a pas été effectuée.");
}

echo "Recherche avec ldap_list()<br />";
$recherche = ldap_list($idLdap, "dc=kangouroo,dc=homelinux,dc=org",
                               "objectclass=*");
$resultat = ldap_get_entries($idLdap, $recherche);
```

```
echo "Il y a ".$resultat["count"]." résultats<br />";
for ($i=0; $i<$resultat["count"]; $i++)
{
    echo $resultat[$i]['dn']."<br />";
}

echo "<br />";
echo "Recherche avec ldap_search()<br />";
$recherche = ldap_search($idLdap, "dc=kangouroo,dc=homelinux,dc=org",
                                  "objectclass=*");
$resultat = ldap_get_entries($idLdap, $recherche);

echo "Il y a ".$resultat["count"]." résultats<br />";
for ($i=0; $i<$resultat["count"]; $i++)
{
    echo $resultat[$i]['dn']."<br />";
}
echo "<br />";

echo "Recherche avec ldap_read()<br />";
$recherche = ldap_read($idLdap, "dc=kangouroo,dc=homelinux,dc=org",
                                "objectclass=*");
$resultat = ldap_get_entries($idLdap, $recherche);

echo "Il y a ".$resultat["count"]." résultats<br />";
for ($i=0; $i<$resultat["count"]; $i++)
{
    echo $resultat[$i]['dn']."<br />";
}

ldap_close($idLdap);
?>
</body>
</table>
```

Vous devez retrouver plus de résultats avec la fonction `ldap_search()`, car celle-ci va scanner les entrées fils. La fonction `ldap_list()`, quant à elle, ne vous retournera que les réponses qui se trouvent directement en dessous de l'entrée que vous avez spécifiée. `ldap_read()` ne vous retournera qu'un seul résultat.

Récupérer les résultats d'une recherche

Après avoir effectué votre recherche, il faut récupérer les résultats. Là encore, plusieurs fonctions sont mises à votre disposition. Vous pouvez, à l'aide de celles-ci, récupérer les résultats de plusieurs façons :

- Récupérer les résultats en une fois ;
- Récupérer les résultats un à un.

Récupérer les résultats en une fois

Pour récupérer les résultats d'une recherche, vous pouvez utiliser l'instruction `ldap_get_entries()`.

ldap_get_entries()

Retourne un tableau contenant tous les éléments, attributs et valeurs pour une recherche.

Syntaxe	`array ldap_get_entries(resource $idLdap, resource $idResultat)`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`$idResultat`	Identifiant sur le résultat de la recherche tel qu'il est retourné par les fonctions `ldap_search()`, `ldap_list()` ou `ldap_read()`.
`retour`	Tableau contenant tous les résultats de la recherche.

Le tableau résultant de l'exécution de l'instruction `ldap_get_attributes()` se présente comme ceci :

```
<?php
$resultat[0]  // est le premier éléments
$resultat[n]  // est l'élément n de votre recherche
$resultat["count"]  // indique le nombre d'éléments
                    // retournés par votre recherche
$resultat[0][dn]  // indique le DN de l'éléments
$resultat[n]["count"]  // Nombre d'attributs pour l'élément n
$resultat[n][0]  // Nom de l'attribut 0
$resultat[n][n]  // Nom de l'attribut n
$resultat[n][attribut]  // Tableau contenant les valeurs
                        // de l'attribut désigné
$resultat[n][attribut][0]  // Valeur 0 de l'attribut
$resultat[n][attribut][n]  // Valeur n de l'attribut
$resultat[n][attribut]["count"]  // Nombre de valeurs pour l'attribut
?>
```

Exécutez le script suivant en l'adaptant à votre annuaire LDAP pour récupérer lisiblement sur l'écran les données du tableau :

```
<?php
// Effectuez ici votre connexion à l'annuaire

// Recherche avec le filtre objectclass=person
$recherche = ldap_list($idLdap, "dc=domaine,dc=com", "objectclass=person");
$resultat  = ldap_get_entries($idLdap, $recherche);
print_r($resultat);

// Fermez votre connexion à l'annuaire
?>
```

Voici maintenant un exemple de connexion à un annuaire LDAP et d'affichage des résultats. Ici, nous récupérons les données provenant d'un serveur ILS, qui est un annuaire d'utilisateurs de Netmeeting (logiciel de visioconférence), et nous affichons le résultat dans un tableau.

Listing 11.6 : netmeeting.php

```
<html>
    <head>
        <title>Recherche dans un annuaire netmeeting</title>
    </head>
<body>

<?php
// Connexion à l'annuaire
$idLdap = ldap_connect("ils.advalvas.be", 389);
if (!ldap_bind($idLdap)){
    die("La connexion n'a pas été effectuée.");
}

$recherche = ldap_list($idLdap, "objectclass=rtperson",
                "(&(cn=%)(objectclass=rtperson))");
$resultat = ldap_get_entries($idLdap, $recherche);
?>
<table border="0" width="100%">
    <tr>
        <td colspan="5">
            Il y a <?php echo $resultat["count"];?> résultats
        </td>
    </tr>
    <tr>
        <td colspan="5">
            <hr />
        </td>
    </tr>
    <tr>
        <td>
            Nom
        </td>
        <td>
            Prénom
        </td>
        <td>
            ville
        </td>
        <td>
            mail
        </td>
        <td>
            Commentaire
        </td>
    </tr>
<?php
```

```
for ($i=0; $i<$resultat["count"]; $i++)
{
    echo "<tr>";
    echo "  <td>".$resultat[$i]['cn'][0]."</td>";
    echo "  <td>".$resultat[$i]['surname'][0]."</td>";
    echo "  <td>".$resultat[$i]['location'][0]."</td>";
    echo "  <td>".$resultat[$i]['rfc822mailbox'][0]."</td>";
    echo "  <td>".$resultat[$i]['comment'][0]."</td>";
    echo "</tr>";
}
?>
</table>
<?
ldap_close($idLdap);
?>
</body>
</table>
```

L'ILS n'est pas un annuaire LDAP

Les serveurs d'annuaires de visioconférences ne sont pas des annuaires LDAP. Ceux-ci possèdent un schéma particulier permettant au logiciel de visioconférence de s'enregistrer dessus et d'effectuer, toutes les x secondes, des connexions maintenant l'entrée dans l'annuaire. Dans le cas contraire, celle-ci est supprimée. L'interrogation utilise le protocole LDAP, ce qui nous permet d'utiliser PHP pour interroger le serveur.

Vous pouvez, à l'aide d'OpenLDAP, vous construire un annuaire LDAP. Le "howto" (en anglais) que vous trouverez sur la page **http://www.freesoft.org/software/NetMeeting/** *vous explique comment modifier l'annuaire afin qu'il supporte le protocole très particulier de Netmeeting.*

Récupérer les résultats un à un

L'affichage de la recherche élément par élément est composé de plusieurs étapes :

- Création d'un pointeur sur le premier élément de notre recherche ;
- Récupération des données de cet élément ;
- Déplacement du pointeur sur l'élément suivant.

Retourner un identifiant sur le premier élément

Pour récupérer l'identifiant du premier élément de notre recherche, nous utilisons la fonction `ldap_first_entry()`.

ldap_first_entry()

Retourne un pointeur sur le premier élément résultant de notre recherche.

Syntaxe	`resource ldap_first_entry(resource $idLdap, resource $idResultat)`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`$idResultat`	Identifiant sur le résultat de la recherche tel qu'il est retourné par les fonctions `ldap_search()`, `ldap_list()` ou `ldap_read()`.
`retour`	Identifiant sur le premier élément de la recherche. Retourne `FALSE` en cas d'erreur ou lorsque la recherche n'a retourné aucun résultat.

Récupération des données d'un élément

La récupération des résultats d'un élément se fait par l'appel à la fonction `ldap_get_attributes()`.

ldap_get_attributes()

Retourne les attributs et leurs valeurs composant l'élément.

Syntaxe	`array ldap_get_attributes(resource link_identifier, resource $idEntree)`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`$idEntree`	Identifiant sur l'élément tel qu'il est retourné par l'instruction `ldap_first_entry()` ou `ldap_next_entry()`.
`retour`	Tableau contenant les différents attributs et les valeurs composant l'élément.

Déplacer le pointeur sur l'élément suivant

Le pointeur est déplacé sur l'élément suivant par l'utilisation de l'instruction `ldap_next_entry()`.

ldap_next_entry()

Déplace le pointeur sur l'élément suivant de la recherche.

Syntaxe	`resource ldap_next_entry(resource $idLdap, resource $idEntree)`

`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`$idEntree`	Identifiant sur l'élément tel qu'il est retourné par l'instruction `ldap_first_entry()` ou `ldap_next_entry()`.
`retour`	Identifiant sur l'élément suivant de la recherche, ou `FALSE` en cas d'erreur ou lorsque la recherche ne trouve plus aucun résultat.

Exemple de recherche

Listing 11.7 : ldap_get_attributes.php

```
<html>
<body>

<?php
// Connexion à l'annuaire
$idLdap = ldap_connect("localhost",389);
// Authentification sur l'annuaire
if (!ldap_bind($idLdap)){
    die("La connexion n'a pas été effectuée.");
}

// Recherche avec le filtre objectclass=person
$recherche = ldap_list($idLdap, "dc=domaine,dc=com", "objectclass=person");

// On récupère la première entrée
$entree = ldap_first_entry($idLdap, $recherche);

// Tant qu'il y a des réponses on récupère les différents attributs
while($entree){
    $resultat = ldap_get_attributes($idLdap, $entree);

    // Boucle suivant le nombre d'attributs trouvés
    // dans la recherche
    for($i=0; $i<$resultat['count']; $i++ ){
        // Affichage de l'attribut
        echo "$resultat[$i]=";
        // Affichage du résultat, la boucle est utile si l'attribut
        // possède de multiples valeurs
        for( $j=0; $j<$resultat[$resultat[$i]]['count'];$j++)
        {
            echo $resultat[$resultat[$i]][$j]." ";
        }
    }
    echo "<hr />";

    // On passe à l'attribut suivant
    $entree = ldap_next_entry($idLdap, $entree);
}
```

```
// Fermeture de la connexion avec l'annuaire
ldap_close($idLdap);
?>
</body>
</table>
```

Autre méthode

Nous pouvons également récupérer les attributs les uns après les autres pour chaque élément retourné par notre recherche. Pour cela, après avoir récupéré le premier élément, nous faisons appel à l'instruction `ldap_first_attribute()`.

ldap_first_attribute()

Récupère le nom du premier attribut d'un élément.

Syntaxe	`string ldap_first_attribute(resource $idLdap, resource $idEntree, resource &$idAttribut)`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`$idEntree`	Identifiant sur l'élément tel qu'il est retourné par l'instruction `ldap_first_entry()` ou `ldap_next_entry()`.
`$idAttribut`	Identifiant passé par référence. Utilisé par la suite par `ldap_next_attribute()`.
`retour`	Nom du premier attribut de l'élément, ou `FALSE` en cas d'erreur.

Pour récupérer les différents attributs suivants, il faut utiliser l'instruction `ldap_next_attribute()`.

ldap_next_attribute()

Récupère les noms des différents attributs d'un élément.

Syntaxe	`string ldap_next_attribute(resource $idLdap, resource $idEntree, resource &$idAttribut)`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`$idEntree`	Identifiant sur l'élément tel qu'il est retourné par l'instruction `ldap_first_entry()` ou `ldap_next_entry()`.
`$idAttribut`	Identifiant passé par référence. Chacun des appels à cette instruction fait avancer ce pointeur sur l'attribut suivant.
`retour`	Nom du premier attribut de l'élément, ou `FALSE` en cas d'erreur.

Nous pouvons ensuite récupérer les différentes valeurs de l'attribut en appelant les fonctions `ldap_get_values()` ou `ldap_get_values_len()`.

La fonction `ldap_get_values()` récupère les différentes valeurs de l'attribut spécifié.

ldap_get_values()

Retourne la ou les valeurs de l'attribut dans un tableau.

Syntaxe	`array ldap_get_values(resource $idLdap, resource $idEntree, string $attribut)`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`$idEntree`	Identifiant sur l'élément tel qu'il est retourné par l'instruction `ldap_first_entry()` ou `ldap_next_entry()`.
`$attribut`	Nom de l'attribut.
`retour`	Tableau contenant les différentes valeurs.

Le tableau résultant de l'exécution de l'instruction `ldap_get_values()` se présente comme ceci :

```
<?php
$resultat[0]  // première valeur de l'attribut
$resultat[n]  // valeur n de l'attribut
$resultat["count"]  // indique le nombre de valeur de l'attribut
?>
```

Voici un exemple complet qui permet de comprendre comment récupérer les éléments, les attributs et chacune des valeurs correspondantes.

```
<html>
<body>

<?php
// Connexion à l'annuaire
$idLdap = ldap_connect("localhost",389);
// Authentification sur l'annuaire
if (!ldap_bind($idLdap)){
    die("La connexion n'a pas été effectuée.");
}

// Recherche avec le filtre objectclass=person
$recherche = ldap_list($idLdap, "dc=domaine,dc=com", "objectclass=person");

// On récupère la première entrée
$entree = ldap_first_entry($idLdap, $recherche);

// Tant qu'il y a des réponses on récupère les différents attributs
```

```
while($entree){
    // On récupère le premier attribut
    $attribut = ldap_first_attribute($idLdap, $entree, $idReference);
    // Affichage du résultat, la boucle est utile si l'attribut
    // possède de multiples valeurs
    while($attribut){
        // Affichage du nom de l'attribut
        echo $attribut."($idReference)=";
        // Récupération des valeurs de l'attribut
        $valeurs = ldap_get_values($idLdap, $entree, $attribut);
        for($i=0; $i<$valeurs["count"]; $i++)
        {
            // Affichage des différentes valeurs
            echo $valeurs[$i]."($idReference)<br />";
        }
        $attribut = ldap_next_attribute($idLdap, $entree, $idReference);
    }
    echo "<hr />";

    // On passe à l'attribut suivant
    $entree = ldap_next_entry($idLdap, $entree);
}

// Fermeture de la connexion avec l'annuaire
ldap_close($idLdap);
?>
</body>
</table>
```


La fonction `ldap_get_values()` ne permet pas de récupérer les valeurs des attributs binaires. Pour cela, vous devez utiliser l'instruction `ldap_get_values_len()`. Cette fonction retourne tous les éléments, ainsi que les éléments binaires.

ldap_get_values_len()

Récupère la ou les valeurs des attributs binaires.

Syntaxe	`array ldap_get_values_len(resource $idLdap, resource $idEntree, string $attribut)`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`$idEntree`	Identifiant sur l'élément tel qu'il est retourné par l'instruction `ldap_first_entry()` ou `ldap_next_entry()`.
`$attribut`	Nom de l'attribut.
`retour`	Tableau contenant les différentes valeurs.

L'utilisation de l'instruction `ldap_get_values_len()` est identique à celle de l'instruction `ldap_get_values()`.

Ordonner les réponses

Vous pouvez ordonner les réponses en utilisant la fonction `ldap_sort()`.

ldap_sort()

Ordonne les résultats dans l'ordre alphabétique.

Syntaxe	`boolean ldap_sort(resource $idLdap, resource $idResultat, string $filtre)`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`$idResultat`	Identifiant sur le résultat de la recherche tel qu'il est retourné par les fonctions `ldap_search()` et `ldap_list()`.
`$filtre`	Indique quel est l'attribut qui permet d'ordonner les résultats.
`retour`	`TRUE` si les résultats ont bien été ordonnés, `FALSE` dans le cas contraire.

Prenez bien garde à ordonner vos résultats avant de faire appel à la fonction `ldap_get_entries()`, car la fonction n'aurait alors aucun effet.

Compter les résultats d'une recherche

Il existe deux façons de déterminer le nombre d'enregistrements retournés par une recherche dans l'annuaire.

Si vous avez effectué une recherche et récupéré les résultats à l'aide de la fonction `ldap_get_entries()`, vous pouvez simplement retrouver le nombre d'entrées depuis le tableau qui vous est retourné de la façon suivante : `$tableau["count"]`.

```
<?php
$recherche = ldap_list($idLdap,"dc=domaine,dc=com ","objectclass=person");
$resultat = ldap_get_entries($idLdap, $recherche);
// Affichage du nombre d'entrées
echo "Il y a ".$resultat["count"]." entrées pour votre recherche !";
?>
```

Si vous avez utilisé l'instruction `ldap_get_attributes()`, alors vous ne pouvez retrouver le nombre d'éléments retournés par votre recherche depuis le tableau de résultat (mais seulement le nombre d'attributs). Dans ce cas, vous pouvez utiliser la fonction :

ldap_count_entries()

Retourne le nombre de résultats d'une recherche.

Syntaxe	`int ldap_count_entries(resource $idLdap, resource $idResultat)`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`$idResultat`	Identifiant sur le résultat de la recherche tel qu'il est retourné par les fonctions `ldap_search()`, `ldap_list()` ou `ldap_read()`.
`retour`	Entier indiquant le nombre de résultats de la recherche, ou `FALSE` en cas d'erreur.

Comparer un attribut avec une valeur

Plus rapide que d'effectuer une recherche, vous pouvez simplement comparer la valeur d'un attribut avec une donnée. Cela vous permet, par exemple, de vérifier un mot de passe dans l'annuaire. On utilise, pour cela, l'instruction `ldap_compare()`.

ldap_compare()

Compare la valeur d'un attribut avec une chaîne passée en paramètre. À noter que l'instruction est insensible à la casse des caractères.

Syntaxe	`boolean ldap_compare(resource $idLdap, string $dn, string $attribut, string $valeur)`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`$dn`	DN de l'entrée à vérifier.
`$attribut`	Attribut qui contient la chaîne à comparer.
`$valeur`	Chaîne de caractères à comparer avec `$attribut`.
`retour`	`TRUE` si les valeurs sont identiques, `FALSE` dans le cas contraire.

Libérer la mémoire

Si vous effectuez des recherches lourdes et successives, vous pouvez avoir des problèmes de mémoire. Par défaut, lorsque vous fermez votre connexion, la mémoire est automatiquement libérée. Mais, dans le cas où votre programme effectue des requêtes successives, vous serez alors amené à libérer la mémoire entre chaque transaction. La fonction `ldap_free_result()` libère la mémoire en vidant les résultats précédents.

ldap_free_result()

Vide la mémoire et efface les résultats obtenus précédemment.

Syntaxe	`boolean ldap_free_result(resource $idLdap)`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`retour`	`TRUE` si la mémoire a été libérée, `FALSE` dans le cas contraire.

Gestion des erreurs

Trois fonctions permettent de retrouver les différents messages d'erreur qui peuvent survenir. Ces fonctions sont `ldap_errno()`, `ldap_error()` ou `ldap_err2str()`.

ldap_errno()

Récupère le numéro de la dernière erreur rencontrée par LDAP.

Syntaxe	`int ldap_errno(resource $idLdap)`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`retour`	Entier indiquant le numéro de l'erreur. `0` indique que l'opération a été effectuée avec succès.

ldap_error()

Récupère le message de la dernière erreur rencontrée par LDAP.

Syntaxe	`string ldap_error(resource $idLdap)`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`retour`	Message d'erreur.

Vous pouvez retrouver ce message d'erreur depuis le numéro retourné par `ldap_errno()` simplement en faisant appel à l'instruction `ldap_err2str()`.

ldap_err2str()

Retourne le message associé au numéro de l'erreur.

Syntaxe	`string ldap_err2str(int $numErreur)`
`$numErreur`	Numéro de l'erreur.
`retour`	Message d'erreur.

Pour consulter les différents messages possibles, lancez le script suivant :

```
<?php
for($i=0;$i<100;$i++)
{
    $message = ldap_err2str($i);
    echo "$i :   ";
    echo $message."<br />";
}
?>
```


Opération sur le Distinguished Name (DN)

Récupérer le DN

Nous avons vu que, lorsque nous effectuons une recherche et que nous récupérons les résultats par l'instruction `ldap_get_entries()`, nous pouvons récupérer le DN simplement depuis le tableau retourné : `$tableau[$n]["dn"]`. Si nous récupérons les résultats un à un avec la fonction `ldap_get_attributes()`, nous devrons avoir recours à l'instruction `ldap_get_dn()`.

ldap_get_dn()

Permet de récupérer le DN par rapport à un résultat de l'entrée.

Syntaxe	`string ldap_get_dn(resource $idLdap, resource $idEntree)`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`$idResultat`	Identifiant sur l'entrée tel qu'il est retourné par les fonctions `ldap_first_entry()`.
`retour`	Le DN (Distinguished Name).

Conversion de format

Une fonction particulière permet de convertir le DN dans un format plus convivial, l'UFN (User Friendly Naming). Ce format enlève les noms des RDN (Relative Distinguished Name) du DN, et ne retourne que leurs valeurs séparées par des virgules.

ldap_dn2ufn()

Nettoie le DN des noms des RDN qui le composent.

Syntaxe	`string ldap_dn2ufn(string $dn)`
`$dn`	DN à convertir.
`retour`	Chaîne de caractères contenant le DN sans les noms des RDN.

```
<?php
$dn = "cn=Laurent GUEDON,ou=développement, o=tild, c=france";
echo ldap_dn2ufn($dn);
?>
Laurent GUEDON,développement, tild, france
```

Récupérer les composantes du DN

La fonction `ldap_explode_dn()` permet de récupérer les différentes valeurs des RDN (Relative Distinguished Name) du DN, accompagné ou non de l'identifiant de ces RDN.

ldap_explode_dn()

Retourne un tableau contenant les différentes composantes du DN.

Syntaxe	`array ldap_explode_dn(string $dn, int $sansRdn)`
`$dn`	DN à convertir.
`$sansRdn`	Indique de retourner uniquement les valeurs des RDN sans leurs identifiants.
`retour`	Tableau contenant le DN ainsi découpé.

```
<?php
$dn = "cn=Laurent GUEDON,ou=développement, o=tild, c=france";
echo "affiche le tableau avec les noms des RDN\n";
$tableau = ldap_explode_dn($dn, 0);
print_r($tableau);
echo "\n";
echo "affiche le tableau sans les noms des RDN\n";
$tableau = ldap_explode_dn($dn, 1);
```

```
print_r($tableau);
?>
affiche le tableau avec les noms des RDN
Array
(
    [count] => 4
    [0] => cn=Laurent GUEDON
    [1] => ou=développement
    [2] => o=tild
    [3] => c=france
)

affiche le tableau sans les noms des RDN
Array
(
    [count] => 4
    [0] => Laurent GUEDON
    [1] => développement
    [2] => tild
    [3] => france
)
```


Opération sur les options

Modifier une option LDAP

Pour modifier une option LDAP, nous utiliserons la fonction `ldap_set_option()`.

ldap_set_option()

Modifie une option de l'annuaire LDAP.

Syntaxe	`boolean ldap_set_option(resource $idLdap, int $option, mixed $nouvelleValeur)`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`$option`	Identifiant de l'option à modifier.
`$nouvelleValeur`	Nouvelle valeur à donner à l'option.
`retour`	`TRUE` en cas de succès du changement de l'option, `FALSE` dans le cas contraire.

Tableau 11.1 : Les différentes options modifiables par ldap_set_option()

Option	Description
LDAP_OPT_DEREF	Ce paramètre indique comment les alias des attributs doivent être considérés. Utilisez l'une des constantes suivantes : *LDAP_DEREF_NEVER*, indique que les alias peuvent être utilisés pour effectuer la recherche. *LDAP_DEREF_SEARCHING*, indique que les alias ne peuvent pas être utilisés pour la recherche, mais que, lors du renvoi des résultats, ceux-ci sont utilisés. *LDAP_DEREF_FINDING*, indique que les alias sont utilisés pour la recherche, mais que le résultat ne les prend pas en compte. *LDAP_DEREF_ALWAYS*, indique que les alias ne sont jamais pris en compte, ni pour la recherche ni pour le résultat.
LDAP_OPT_SIZELIMIT	Indique le nombre de résultats maximal qui peut être retourné. La valeur par défaut `0` (*LDAP_NO_LIMIT*) indique qu'il n'y a pas de limite maximale.
LDAP_OPT_TIMELIMIT	Indique le temps maximal d'exécution d'une requête dans l'annuaire LDAP. Cette valeur affecte simplement la recherche sur le serveur. La valeur par défaut `0` (*LDAP_NO_LIMIT*) indique qu'il n'y a pas de limite de temps.
LDAP_OPT_REFERRALS	Cette option indique comment le client doit réagir lorsqu'une référence lui est retournée par le serveur. Le paramètre par défaut est *LDAP_OPT_ON*, c'est-à-dire que les références sont prises en compte. *LDAP_OPT_OFF* empêche cela.
LDAP_OPT_RESTART	Indique si, lors d'une interruption prématurée (comme un time out par exemple), les opérations interrompues doivent reprendre (*LDAP_OPT_ON*) ou non (*LDAP_OPT_OFF*). Par défaut, l'option est fixée à *LDAP_OPT_ON*.
LDAP_OPT_PROTOCOL_VERSION	Indique la version du protocole LDAP à utiliser : *LDAP_VERSION2* pour la version 2 ou *LDAP_VERSION3* pour la version 3. Par défaut, le protocole utilisé est v2.

LDAP_OPT_SERVER_CONTROLS	Indique une liste de contrôles au niveau du serveur. Le serveur doit les recevoir à chaque fois. La valeur qui doit être passée à l'instruction est un tableau contenant les différents contrôles.
	`array(`
	`  array( // premier contrôle`
	`  "oid" => "x.x.x.x.x.x",`
	`  "iscritical" => TRUE,`
	`  ),`
	`  array( // deuxième contrôle`
	`  "oid" => "y.y.y.y.y.y",`
	`  "iscritical" => FALSE,`
	`  ),`
	`)` Par défaut, il n'y aucun contrôle au niveau du serveur.

LDAP_OPT_CLIENT_CONTROLS	Listes de contrôles au niveau du client.

Consulter une option LDAP

Pour récupérer une option du serveur LDAP, on utilisera la fonction :

ldap_get_option()

Récupère une option de l'annuaire LDAP.

Syntaxe	`boolean ldap_get_option(resource $idLdap, int $option, mixed &$valeurRetour)`
`$idLdap`	Identifiant sur la connexion au serveur LDAP tel qu'il est retourné par la fonction `ldap_connect()`.
`$option`	Identifiant de l'option à modifier.
`$valeurRetour`	Variable dans laquelle sera copiée la valeur de l'option que l'on aura récupérée.
`retour`	`TRUE` en cas de succès, `FALSE` dans le cas contraire.

Vous pouvez récupérer les mêmes options que pour la fonction `ldap_set_option()`, plus quelques autres en lecture seule.

Tableau 11.2 : Les options supplémentaires de la fonction ldap_get_option()

Option	Description
LDAP_OPT_ERROR_NUMBER	Retourne le numéro de la dernière erreur rencontrée. Identique à la fonction `ldap_errno()`.
LDAP_OPT_ERROR_STRING	Retourne la dernière erreur rencontrée. Identique à la fonction `ldap_error()`.
LDAP_OPT_HOST_NAME	Retourne l'adresse et le port de la connexion au serveur LDAP.

Vous trouverez diverses explications (en anglais) sur les options LDAP sur le site de NETSCAPE : **http://developer.netscape.com/docs/manuals/dirsdk/csdk30/functi52.htm**.

11.4. Exemple d'application

Pour illustrer ce chapitre, nous allons créer un petit annuaire couplé avec un navigateur LDAP. Ce n'est pas là l'une des grosses applications que vous pouvez trouver dans une entreprise pour gérer le personnel et les clients, mais les bases sont là. À vous de faire évoluer cet annuaire par la suite afin d'intégrer de nouvelles fonctions.

L'authentification sur l'annuaire

Il est simple à présent de construire un petit système d'authentification sur un annuaire. Dans un premier temps, nous allons préparer un fichier de configuration qui contiendra tout paramètre propre à notre annuaire LDAP.

Listing 11.8 : exemples/configuration.php

```
<?php
/*
/ Paramètres du serveur LDAP
*/
$serveur = "localhost";     // Adresse du serveur LDAP
$port = 389;                // Port du serveur LDAP
$ssl = 0;                   // Utilisez 1 pour une connexion par SSL

// DN Root
$dnOrigine = "dc=mondomaine,dc=com";

// Branche utilisateur racine
$dnUtilisateur = "ou=People,dc=mondomaine,dc=com";

// Attribut login d'un utilisateur
$loginAttribut = "cn";

// dn du manager de l'annuaire LDAP
$dnRoot = "cn=root,dc=mondomaine,dc=com";


/*
/ Paramètres des pages
*/
$titre        = "Bldap - L'annuaire LDAP en PHP";
$couleurBoite = "#0066CC";
$couleurText  = "#FFFFFF";
?>
```

Nous pouvons, à présent, écrire un exemple d'authentification. Ceci commence par la page d'index qui gère les sessions des utilisateurs une fois authentifiées :

Listing 11.9 : exemples/index.php

```
<?php
include("includes/inclusion.php");

session_start();

if ($_POST["action"])
{
    $action = $_POST["action"];
} elseif ($_GET["action"]){
```

```
    $action = $_GET["action"];
} else {
    $action = "";
}

switch ($action)
{
    case "connexion":
        connexion();
        affichehtml();
        break;
    case "deconnexion":
        deconnexion();
        affichehtml();
        break;
    case "recherche":
        include("entete.php");
        afficheRecherche();
        break;
    case "modifier":
        include("entete.php");
        modifierFiche();
        break;
    case "fiche":
        include("entete.php");
        afficheFiche();
        break;
    default:
        include("entete.php");
        if (!isset($_SESSION['sessionConnexionType'])) {
            // Si aucune session n'a été créée
            afficheBoiteLogin();
        } else {
            // Si l'utilisateur est connecté à l'interface
            afficheBlocRecherche();
        }
        break;
}

?>
<p> </p>
<p> </p>

<?php
include("pieddepage.php");
?>
```

Inclusion est un fichier qui contient tous les autres fichiers PHP à inclure. Il ressemble à ceci :

Listing 11.10 : exemples/includes/inclusion.php

```
<?php
require_once("configuration.php");
require_once("html.php");

require_once("includes/connexionLdap.php");
require_once("includes/identification.php");
require_once("includes/ldap.php");
?>
```

Les fichiers *entete.php* et *pieddepage.php* contiennent le code HTML permettant de donner le look général de la page. Le fichier *html.php* contient différentes fonctions affichant les formulaires ou les résultats des réponses. Les fonctions `connexion()` et `deconnexion()` sont décrites dans le fichier *identification.php* qui se trouve dans le répertoire *includes*.

Listing 11.11 : exemples/includes/identification.php

```
<?php
/**
 * Fonction de connexion à l'interface
 * @return $ldap ou FALSE si erreur dans l'identification
 */
function connexion()
{
    global $dnUtilisateur, $loginAttribut, $dnRoot;
    global $motPasse, $utilisateur, $connexionType;

    if ($ldap = connexionLdap())
    {
        if ($connexionType == "normal")
        {
            $_SESSION['sessionDn']              = $loginAttribut.
                        "=".$utilisateur.",".$dnUtilisateur;
            $_SESSION['sessionPass']            = $motPasse;
            $_SESSION['sessionConnexionType']   = $connexionType;
            $_SESSION['sessionRdn']             = $utilisateur;
        } elseif ($connexionType == "manager") {
            $_SESSION['sessionDn']              = $dnRoot;
            $_SESSION['sessionPass']            = $motPasse;
            $_SESSION['sessionConnexionType']   = $connexionType;
            $_SESSION['sessionRdn']             = "Manager";
        } else {
            $_SESSION['sessionDn']              = "";
            $_SESSION['sessionPass']            = "";
            $_SESSION['sessionConnexionType']   = $connexionType;
            $_SESSION['sessionRdn']             = "Anonyme";
        }
    } else {
        return FALSE;
    }
    deconnexionLdap($ldap);
```

```
    afficheBoiteMsg("La connexion a été effectuée");
    return TRUE;
}

/**
 * Fonction de déconnexion de l'interface
 */
function deconnexion()
{
    session_unset();
    session_destroy();
    afficheBoiteMsg("La deconnexion a été effectuée");
}
?>
```


La fonction `connexion()` fait appel à deux fonctions `connexionLdap()` et `deconnexionLdap()` qui sont décrites dans le fichier *includes/connexionLdap.php*. Suivant le résultat de l'authentification LDAP, le script crée une session ou affiche un message indiquant que l'authentification a échoué.

Listing 11.12 : exemples/includes/connexionLdap.php

```
<?php
/**
 * Fonction de création de connexion et d'authentification au serveur LDAP
 * @return resource Identifiant de connexion LDAP ou FALSE si problème
 */
function connexionLdap()
{
    global $serveur, $port, $ssl;
    global $dnUtilisateur, $loginAttribut, $dnRoot;
    global $motPasse, $utilisateur, $connexionType;

    // Connexion au serveur LDAP
    if ($ssl) $serveur = "ldaps://".$serveur;
    $ldap = @ldap_connect($serveur, $port);
    if (!$ldap)
    {
        afficheBoiteMsgErreur("La connexion au ".
                serveur n'a pas été effectuée !");
        return FALSE;
    }

    // Vérification qu'une session n'est pas déjà créée
    if (isset($_SESSION['sessionConnexionType']))
    {
        $sessionDn = $_SESSION['sessionDn'];
        $sessionPass = $_SESSION['sessionPass'];
        $sessionConnexionType = $_SESSION['sessionConnexionType'];
    } else {
        $sessionDn = $loginAttribut."=".$utilisateur.",".$dnUtilisateur;
        $sessionPass = $motPasse;
```

```
        $sessionConnexionType = $connexionType;
    }

    // Connexion sur l'annuaire en mode utilisateur "normal"
    if ($sessionConnexionType=="normal" && $sessionDn && $sessionPass)
    {
        // Authentification sur le serveur LDAP
        if (!@ldap_bind($ldap, $sessionDn, $sessionPass))
        {
            afficheBoiteMsgErreur("L'authentification a échoué !");
            return FALSE;
        }
    }
    // Connexion sur l'annuaire en mode utilisateur "manager"
    elseif ($sessionConnexionType=="manager" && $sessionPass)
    {
        // Authentification sur le serveur LDAP
        if (!@ldap_bind($ldap, $dnRoot, $sessionPass))
        {
            afficheBoiteMsgErreur("L'authentification a échoué !");
            return FALSE;
        }
    }
    // Connexion sur l'annuaire en mode utilisateur "anonyme"
    elseif ($sessionConnexionType=="anonyme")
    {
        // Authentification sur le serveur LDAP
        if (!@ldap_bind($ldap))
        {
            afficheBoiteMsgErreur("L'authentification a échoué !");
            return FALSE;
        }

    } else {
        // Problème de connexion
        afficheBoiteMsgErreur("Connexion impossible !");
        return FALSE;
    }
    return $ldap;
}

/**
 * Fonction de déconnexion du serveur LDAP
 * @input $ldap Identifiant de connexion LDAP
 * @return rien
 */
function deconnexionLdap($ldap)
{
    // Déconnexion du serveur LDAP
    ldap_close($ldap);
}
?>
```

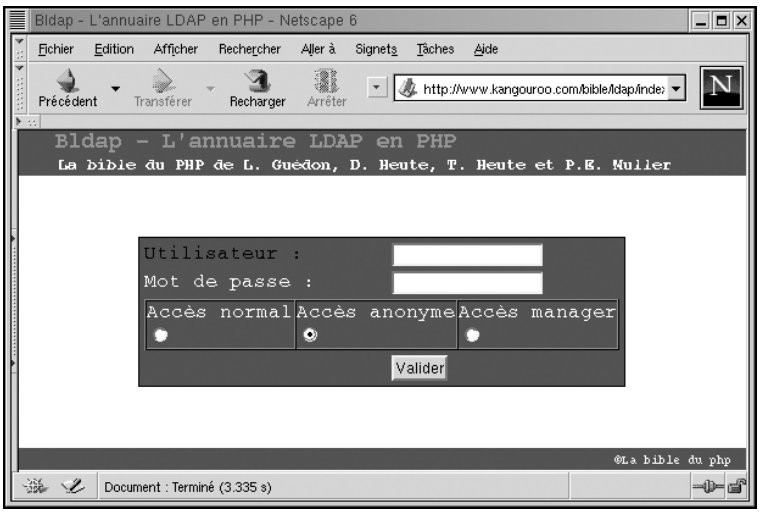


Figure 11.5 : *Vous pouvez maintenant vous authentifier comme utilisateur, manager ou anonyme.*

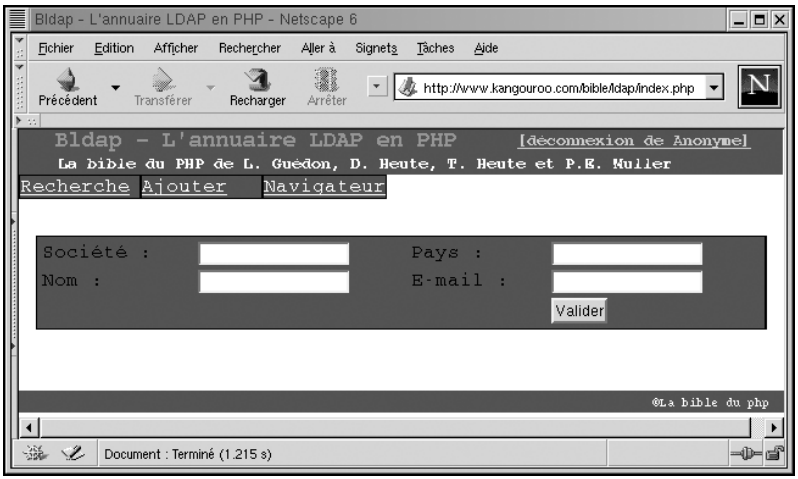


Figure 11.6 : *Vous voilà dans l'interface en tant qu'anonyme*

L'ajout d'une nouvelle entrée

L'ajout d'une nouvelle entrée peut être fait simplement depuis un formulaire HTML. Le schéma de votre annuaire doit supporter les attributs standard que nous utilisons ici. Dans le cas contraire, veuillez vous reporter à la documentation d'OpenLDAP afin de résoudre les éventuels problèmes.

Commençons par afficher la page HTML contenant le formulaire d'ajout d'un nouvel utilisateur.

Listing 11.13 : exemples/ajout.php

```
<?php
include("includes/inclusion.php");

session_start();

if ($_POST["action"])
{
    $action = $_POST["action"];
} elseif ($_GET["action"]){
    $action = $_GET["action"];
} else {
    $action = "";
}


include("entete.php");

switch ($action)
{
    case "ajouter":
        ajouterUtilisateur();
        break;
    default:
        afficheFormulaire();
        break;
}


?>
<p> </p>
<p> </p>

<?php
include("pieddepage.php");
?>
```

La fonction `afficheFormulaire()` est décrite dans le fichier *html.php*. Lorsque le formulaire est validé, le paramètre `action=ajouter` est alors passé au script, ce qui a pour effet d'exécuter la fonction `ajouterUtilisateur()`.

Listing 11.14 : exemples/includes/ldap.php

```
<?php
/**
 * Fonction d'ajout d'un utilisateur dans l'annuaire
 * @return boolean
 */

function ajouterUtilisateur()
```

```
{
    global $loginAttribut, $dnUtilisateur;
    if (!$ldap = connexionLdap()) return FALSE;

    $dn = "$loginAttribut=".$_POST[$loginAttribut].",$dnUtilisateur";

    if ($_POST['objectclass'][0])
        $entree['objectclass'][0]   = $_POST['objectclass'][0];
    if ($_POST['objectclass'][1])
        $entree['objectclass'][1]   = $_POST['objectclass'][1];
    if ($_POST['sn'])
        $entree['sn'] = $_POST['sn'];
    if ($_POST['givenname'])
        $entree['givenname']        = $_POST['givenname'];
    if ($_POST['mail'])
        $entree['mail']             = $_POST['mail'];
    if ($_POST['telephonenumber'])
        $entree['telephonenumber']  = $_POST['telephonenumber'];
    if ($_POST['telexnumber'])
        $entree['telexnumber']      = $_POST['telexnumber'];
    if ($_POST['o'])
        $entree['o']                = $_POST['o'];
    if ($_POST['ou'])
        $entree['ou']               = $_POST['ou'];
    if ($_POST['street'])
        $entree['street']           = $_POST['street'];
    if ($_POST['postalcode'])
        $entree['postalcode']       = $_POST['postalcode'];
    if ($_POST['l'])
        $entree['l']                = $_POST['l'];
    if ($_POST['c'])
        $entree['c']                = $_POST['c'];
    if ($_POST['userpassword'])
        $entree['userpassword']     = $_POST['userpassword'];

        $entree[$loginAttribut]     = $_POST[$loginAttribut];

    if (@ldap_add($ldap, $dn, $entree))
    {
        afficheBoiteMsg("Le nouvel utilisateur a été ajouté !");
    } else {
        afficheBoiteMsgErreur("Problème, le nouvel".
            " utilisateur n'a pas été ajouté !", $ldap);
        return FALSE;
    }

    // Déconnexion du serveur LDAP
    deconnexionLdap($ldap);

    return TRUE;
}
```

Recherche dans l'annuaire

À présent que de nouveaux utilisateurs sont dans l'annuaire, il est grand temps d'effectuer une recherche qui va nous permettre de les retrouver. C'est un formulaire HTML simple qui va nous permettre de rechercher les individus selon plusieurs critères :

- La société qui est représentée par l'attribut `o` ou `organization` ;
- Le pays qui est défini par l'attribut `c` ;
- Le nom qui est désigné par l'attribut `cn` ;
- L'e-mail représenté par l'attribut `mail`.

Le formulaire est appelé depuis notre page d'index par la fonction `afficheBlocRecherche()`. Sa description se trouve dans le fichier *html.php*.

La fonction qui affiche le résultat est `afficheRecherche()`.

Listing 11.15 : exemples/html.php (extrait)

```
<?php
/**
 * Fonction affichage le résultat de la recherche
 */
function afficheRecherche()
{
    global $loginAttribut;

    afficheBlocRecherche($contenu);
?>
    <p> </p>
        <table border="0" cellpadding="1" cellspacing="0"
        width="90%" bgcolor="#000000" align="center">
        <tr>
        <td>
        <table border="0" cellpadding="0" cellspacing="4" width="100%"
         bgcolor="#ffffff">
        <tr>
        <td align="center" width="25%">
            Nom
        </td>
        <td align="center" width="25%">
            Identifiant
        </td>
        <td align="center" width="25%">
            E-mail
        </td>
        <td align="center" width="25%">
            Téléphone
        </td>
        </tr>
<?php
```

```
    $resultat = rechercheLdap();
    echo '<tr><td colspan="4" align="center">'.
                    'Il y a '.$resultat["count"].' résultat(s)'.
                    ' pour votre recherche';
    for ($i=0;$i<$resultat["count"];$i++)
    {
        echo '<tr>';
        echo '<td><a href="index.php?action=fiche&identifiant='.
                $resultat[$i][$loginAttribut][0].'">'.
                $resultat[$i]["cn"][0].'</a></td>';
        echo '<td>'.$resultat[$i]["uid"][0].'</td>';
        echo '<td>'.$resultat[$i]["mail"][0].'</td>';
        echo '<td>'.$resultat[$i]["telephonenumber"][0].'</td>';
        echo '</tr>';
    }
        echo '   </table>
                    </td>
                </tr>
                </table>';
}
?>
```


Enfin, cette fonction fait appel à `rechercheLdap()`, qui s'occupe d'effectuer la recherche dans l'annuaire et de retourner les résultats sous la forme d'un tableau.

Listing 11.16 : exemples/includes/ldap.php

```
<?php
/**
 * Fonction de recherche dans la base LDAP
 * @return array les données recherchées ou FALSE si problème
 */
function rechercheLdap()
{
    global $dnUtilisateur;
    // Connexion au serveur LDAP
    $ldap = connexionLdap();
    if (!$ldap) return FALSE;

    // Création du filtre de recherche
    $filtre = "";
    if ($_POST["societe"])
        $filtre .= "(o=".$_POST["societe"].")";
    if ($_POST["nom"])
        $filtre .= "(cn=".$_POST["nom"].")";
    if ($_POST["pays"])
        $filtre .= "(c=".$_POST["pays"].")";
    if ($_POST["mail"])
        $filtre .= "(mail=".$_POST["mail"].")";
    if (!$filtre)
    {
        $filtre = "(&(cn=*)(objectclass=*))";
```

```
    } else {
        $filtre = "(&".$filtre."(objectclass=*))";

    }
    // Eléments à récupérer
    $element = array("cn", "uid", "mail", "telephonenumber");
    $recherche = @ldap_list($ldap, $dnUtilisateur, $filtre, $element);
    if (!$recherche)
    {
        afficheBoiteMsgErreur("Problème lors de la recherche !", $ldap);
        deconnexionLdap($ldap);
        return FALSE;
    }
    $resultat = @ldap_get_entries($ldap, $recherche);
    deconnexionLdap($ldap);
    return $resultat;
}
?>
```


Modification d'une entrée

Avant de modifier une entrée, nous devons afficher un formulaire. Pour cela, nous allons, dans un premier temps, appeler la page index depuis le résultat de la recherche précédente, et lui transmettre le RDN qui nous permettra ensuite de récupérer l'utilisateur.

La page *index.php* fait alors appel à la fonction `afficheFiche()`.

Listing 11.17 : exemples/html.php (extrait)

```
<?php
/**
 * Fonction affichage de la fiche de la personne
 */
function afficheFiche()
{
?>
        <table border="0" cellpadding="1" cellspacing="0"
        width="90%" bgcolor="#000000" align="center">
        <tr>
        <td>
        <table border="0" cellpadding="0" cellspacing="4" width="100%"
         bgcolor="#ffffff">
            <form action="index.php" method="post">
<?php
    $ldapResultat = rechercheLdapPersonne();

    // Ceci va nous être utile pour bien visualiser les résultats
    // Un peu de couleurs ne fait pas de mal.
    $coul = 0;
    $couleur[0] = "#ffffff";
    $couleur[1] = "#dddddd";
```

```
    for ($i=0;$i<$ldapResultat[0]["count"];$i++)
    {
        $attribut = $ldapResultat[0][$i];

        echo '<tr>';
        echo '<td bgcolor="'.$couleur[$coul].'">'.$attribut.'</td>';

        echo '<td bgcolor="'.$couleur[$coul].'">';

        if ($ldapResultat[0][$i]["count"]){
            // Il peut y avoir plusieurs valeurs pour un attribut.
            // Nous allons donc vérifier combiens de données nous sont
            // retournées pour chaques attributs.
            // Pour cela, on compte le nombre de valeur pour l'attribut
            // et on boucle
            $boucle = $ldapResultat[0][$attribut]["count"];
            if ($boucle>1){
                for ($nbAttr=0;$nbAttr<$boucle;$nbAttr++)
                {
                    echo '<input type="text" name="'.$attribut.
                        '['.$nbAttr.']" value="'.
                        $ldapResultat[0][$attribut][$nbAttr].'"><br />';
                }
            } else {
                echo '<input type="text" name="'.$attribut.
                        '" value="'.
                        $ldapResultat[0][$attribut][0].'">';
            }
        } else {
            echo '<input type="text" name="'.$attribut.'[0]"'.
                        ' value="">';
        }
        echo '<td bgcolor="'.$couleur[$coul].'">';
        echo '</td>';

        echo '</tr>';
        if ($coul==0) $coul=1; else $coul=0;
    }
?>
              <tr><td>
                <input type="hidden" value="modifier" name="action">
                <input type="submit" value="Modifier">
              </td></tr>
            </form>
            </table>
        </td></tr>
    </table>
<?php
}
?>
```

Cette page lance d'abord la fonction `rechercheLdapPersonne()`, qui lui retourne un tableau contenant la liste des attributs et leurs différentes valeurs.

Listing 11.18 : exemples/includes/ldap.php

```
<?php
/**
 * Fonction de récupération des éléments d'une fiche
 * @return array les données recherchées ou FALSE si problème
 */
function rechercheLdapPersonne()
{
    global $dnUtilisateur, $loginAttribut;
    // Connexion au serveur LDAP
    $ldap = connexionLdap();
    if (!$ldap) return FALSE;

    // Création du filtre de recherche
    $filtre = "(&(".$loginAttribut."=".
        $_GET["identifiant"].")(objectclass=*))";

    $recherche = @ldap_list($ldap, $dnUtilisateur, $filtre);
    if (!$recherche)
    {
        afficheBoiteMsgErreur("Problème lors de la recherche !",
                $ldap);
        deconnexionLdap($ldap);
        return FALSE;
    }

    // On récupere l'entrée
    $resultat = @ldap_get_entries($ldap, $recherche);

    // Déconnexion du serveur LDAP
    deconnexionLdap($ldap);

    return $resultat;
}
?>
```

Ici, pas de surprise particulière. C'est une recherche classique dans un annuaire, où l'on précise simplement, en plus, comme filtre, le RDN de l'utilisateur.

Par la suite, la fonction `afficheFiche()` prend le relais et construit le code HTML de notre formulaire en fonction des attributs et de la (ou des) valeur(s) de ceux-ci.

L'envoi du formulaire effectue l'appel depuis la page *index.php* à la fonction `modifierFiche()`.

Listing 11.19 : exemples/includes/ldap.php

```
<?php
/**
 * Fonction de modification d'un utilisateur dans l'annuaire
 * @return boolean
 */

 function modifierFiche()
 {
    global $loginAttribut, $dnUtilisateur;
    if (!$ldap = connexionLdap()) return FALSE;

    $dn = "$loginAttribut=".$_POST[$loginAttribut].",$dnUtilisateur";

    $entree = $_POST;
    array_pop($entree);

    if (@ldap_mod_replace($ldap, $dn, $entree))
    {
        afficheBoiteMsg("L'utilisateur a été modifié !");
    } else {
        afficheBoiteMsgErreur("Problème, ".
                "l'utilisateur n'a pas été modifié !", $ldap);
        return FALSE;
    }

    // Déconnexion du serveur LDAP
    deconnexionLdap($ldap);
    return TRUE;
}
```

Les informations du formulaire sont récupérées et préparées pour être utilisées avec l'instruction `ldap_mod_replace()`. Nous avions pris soin de construire notre formulaire de façon à ce que le tableau `$_POST` soit directement utilisable par cette fonction. La préparation est alors très simple : il suffit de supprimer `$_POST["action"]` à l'aide de l'instruction `array_pop()`.

Réalisation d'un arbre de navigation LDAP

Il est courant de visualiser l'annuaire LDAP sous la forme d'un arbre. Le langage PHP va nous permettre de réaliser un système de navigation simple sur le serveur LDAP.

Listing 11.20 : exemples/navframearbre.php

```
<?php
include("includes/inclusion.php");

session_start();
```

```
affichehtml();
?>
<?php
    if ($_GET['dnEn']){
        $dnEn = urldecode($_GET['dnEn']);
        $dnEnTableau = explode('||', $dnEn);
    } else {
        $dnEnTableau[0] = "";
    }
    echo "<p style='font-size: 10px;'>";
    navigateur($dnEnTableau);
    echo "</p>";
?>
<p> </p>
</body>
</html>
```

Nous observons que la page récupère un paramètre `dnEn` qui lui est envoyé et le transforme en tableau. Ce paramètre va nous servir à ouvrir ou fermer les branches de l'arbre LDAP.

La fonction permettant de visualiser l'arbre est la fonction `navigateur()`.

Listing 11.21 : exemples/includes/ldap.php

```
<?php
/**
 * Fonction affichage de l'élément d'origine de l'arbre LDAP
 * @input $dnEn
 */
function navigateur($dnEn)
{
    global $dnOrigine;
    // Connexion au serveur LDAP
    $ldap = connexionLdap();

    echo "* ";
    // lien vers les attributs
    $lien = "navframedescription.php?dn=".urlencode($dnOrigine);
    echo "<a href='".$lien."' target='description'>".
        $dnOrigine."</a><br />\n";

    extractionDonnee($dnOrigine, $dnEn, $ldap);

    // Déconnexion du serveur LDAP
    deconnexionLdap($ldap);
}
?>
```

La fonction effectue une connexion sur le serveur LDAP, puis affiche les liens sur le premier élément de l'arbre, permettant de visualiser par la suite, sur une nouvelle page, ses attributs. Ensuite, il est fait appel à une nouvelle fonction, `extractionDonnee()`, avant de se déconnecter de l'annuaire.

Listing 11.22 : exemples/includes/ldap.php

```
<?php
/**
 * Fonction récursive permettant de construire l'arbre LDAP
 * @input string $dn, array $dnEn, resource $ldap
 * @return $string données formatés
 */
function extractionDonnee($dn, $dnEn, $ldap)
{
    // Création du filtre de recherche
    $filtre = "(&(objectclass=*))";
    $recherche = @ldap_list($ldap, $dn, $filtre, array(""));
    // On récupere l'entrée
    $tableau = @ldap_get_entries($ldap, $recherche);
    for ($i=0;$i<$tableau["count"];$i++)
    {
        // On récupère le rdn
        $dntmp = split(",", $tableau[$i]["dn"]);

        for($j=0;$j<count($dntmp);$j++)
        {
            echo " ";
        }

        // Les liens sur l'arbre
        $dnEnLien = $dnEn;
        // On vérifie si l'entrée n'existe pas déjà dans le tableau
        if ($clef = array_search($tableau[$i]["dn"], $dnEnLien))
        {
            // Si oui on la supprime
            $dnEnLien[$clef]='';
            $caractere = '-';
        } else {
            // Si non, on l'ajoute
            $dnEnLien[count($dnEn)] = $tableau[$i]["dn"];
            $caractere = '+';
        }

        $chainednTmp = join("||", $dnEnLien);
        // On peut supprimer les signes | en trop
        $chainedn = str_replace("||||","||",$chainednTmp);
        // On encode la chaîne à cause de la methode GET utilisée
        $chainedn = urlencode($chainedn);
        $lien = "navframearbre.php?dnEn=".$chainedn;

        echo "<a href='".$lien."' target='arbre'>[".$caractere."]</a> ";

        // lien vers les attributs
        $lien = "navframedescription.php?dn=".urlencode($tableau[$i]["dn"]);
        echo "<a href='".$lien."' target='description'>".
        $dntmp[0]."</a><br />\n";
```

```
        for ($j=0;$j<count($dnEn);$j++)
        {
            if ($dnEn[$j]==$tableau[$i]["dn"])
            {
                extractionDonnee($tableau[$i]["dn"], $dnEn, $ldap);
            }
        }
    }
}
?>
```

Cette fonction est une fonction récursive. Elle va scanner la branche de l'annuaire et vérifier que le DN ne se trouve pas dans le tableau passé en paramètre. Si elle le trouve, la fonction se lance elle-même avec ce nouveau DN comme nouvelle base de recherche. Les liens nous servent, d'une part, à ouvrir et fermer les branches de l'arbre et, d'autre part, à afficher une page indiquant tous les attributs et leurs valeurs. L'affichage de la page est exécuté par l'appel à la fonction `afficheAttribut()`.

Listing 11.23 : exemples/html.php (extrait)

```
<?php
/**
 * Fonction d'affichage des attributs d'une entrée
 * @input $dn
 */
function afficheAttribut($dn)
{
    $resultat = NavLdapEntree($dn);

    echo '
    <table border="0" cellpadding="1" cellspacing="0"
    width="90%" bgcolor="'.$GLOBALS["couleurBoite"].'" align="center">
    <tr>
    <td>
    <table border="0" cellpadding="0" cellspacing="4" width="100%"
     bgcolor="#ffffff">';

    // Ceci va nous être utile pour bien visualiser les résultats
    // Un peu de couleurs ne fait pas de mal.
    $coul = 0;
    $couleur[0] = "#ffffff";
    $couleur[1] = "#dddddd";

    for ($i=0;$i<$resultat[0]["count"];$i++)
    {
        $attribut = $resultat[0][$i];

        echo '<tr>';
        echo '<td bgcolor="'.$couleur[$coul].'">'.$attribut.'</td>';

        echo '<td bgcolor="'.$couleur[$coul].'">';
```

```
            // Il peut y avoir plusieurs valeurs pour un attribut.
            // Nous allons donc vérifier combiens de données nous sont
            // retournées pour chaques attributs.
            // Pour cela, on compte le nombre de valeur pour l'attribut
            // et on boucle
            for ($nbAttr=0;$nbAttr<$resultat[0][$attribut]["count"];$nbAttr++)
            {
                echo $resultat[0][$attribut][$nbAttr].'<br />';
            }

            echo '<td bgcolor="'.$couleur[$coul].'">';
            echo '</td>';

            echo '</tr>';
            if ($coul==0) $coul=1; else $coul=0;
        }
        echo '</table>
          </td>
          </tr>
        </table>';
}
?>
```

Cette fonction est similaire à celle nous affichant la fiche d'un utilisateur. Elle utilise l'instruction `NavLdapEntree()` en lui envoyant le DN de l'entrée dont nous souhaitons récupérer les attributs.

Listing 11.24 : exemples/includes/ldap.php

```
<?php
/**
 * Fonction de recherche des attributs d'un dn
 * @input $dn
 * @return array les données recherchées ou FALSE si problème
 */
function NavLdapEntree($dn)
{
    // Connexion au serveur LDAP
    $ldap = connexionLdap();
    if (!$ldap) return FALSE;

    // Création du filtre de recherche
    $recherche = @ldap_read($ldap, $dn, "objectclass=*");
    if (!$recherche)
    {
        BoiteMsgErreur("La lecture des informations".
                    " a échoué !", $ldap);
        deconnexionLdap($ldap);
        return FALSE;
    }
```

```
    // On récupere l'entrée
    $resultat = @ldap_get_entries($ldap, $recherche);

    // Déconnexion du serveur LDAP
    deconnexionLdap($ldap);

    return $resultat;
}
?>
```

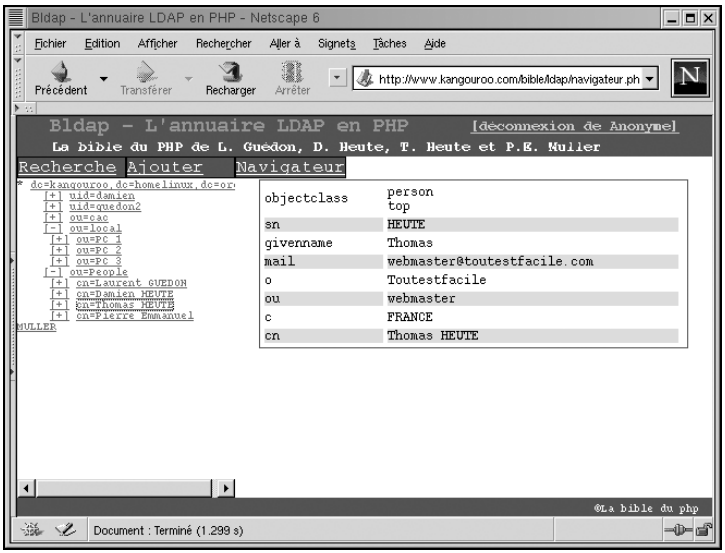


Figure 11.7 : *La navigation simple dans l'annuaire LDAP*

Chapitre 12

La messagerie : envoi et lecture de mails

Envoyer des e-mails ou créer une interface permettant la consultation d'une boîte à lettres électronique sont deux tâches facilitées par l'existence des bibliothèques fournies par PHP.

12.1. E-mail

L'interactivité avec les visiteurs d'un site Internet passe aussi par le courrier électronique. Envoyer un e-mail à un visiteur (ou client) quand un produit susceptible de l'intéresser est disponible ou se faire envoyer (en tant qu'administrateur d'un site) un courrier lorsqu'un message est posté dans le forum sont des exemples typiques d'utilisation du courrier électronique.

Installation

Il y a relativement peu de choses à faire pour profiter des fonctions de mail.

Sous Windows

Il vous suffit de modifier le fichier *php.ini* afin d'ajouter ou décommenter les lignes suivantes :

```
[mail function]
SMTP = smtp.monfai.com
sendmail_from = moi@monfai.com
```

Le paramètre `SMTP` doit préciser l'adresse de votre serveur SMTP (généralement fourni par votre fournisseur d'accès ou votre hébergeur, il peut être du genre *smtp.monfai.com* ou *mail .monfai.com* par exemple).

Le paramètre `sendmail_from` doit préciser votre adresse e-mail.

Commentaires et tabulations
Vous risquez de rencontrer des problèmes si vous conservez les commentaires et tabulations parfois présents sur les lignes de ces paramètres. N'hésitez donc pas à les supprimer.

Si besoin vous pouvez modifier le port d'appel du service SMTP via le paramètre

```
smtp_port = 25
```

Sous Linux

Avec la plupart des distributions, par défaut, vous n'aurez probablement rien à faire.

S'il s'avérait toutefois que cela ne fonctionne pas avec un test simple (le premier exemple présenté), assurez-vous que votre environnement contient bien un client mail appelé `sendmail` ou un équivalent comme `qmail`.

Si ce n'est pas le cas, vous devrez commencer par en installer un (pour cela, consultez la documentation liée à votre distribution). Une fois cette installation faite, vous aurez à recompiler PHP (sans nouvelle option particulière).

Maintenant que vous êtes certain de posséder un client mail (qui fonctionne), sachez que, s'il s'agit de `sendmail`, tout devrait fonctionner automatiquement. Vous pouvez néanmoins préciser le nom (s'il est dans le `PATH`) ou le chemin complet vers votre client mail, ainsi que les options à utiliser avec l'option `sendmail_path` de *php.ini*. Si ce paramètre n'est pas précisé, la valeur par défaut utilisée est "`sendmail -t -i`".

Comme dans l'exemple suivant :

```
[mail function]
sendmail_path = /usr/sbin/sendmail -t -i
```


ATTENTION

Contraintes des serveurs mails

Il est possible que votre serveur mail (notamment s'il est mis à disposition par un hébergeur) vous impose de préciser une adresse retour en cas d'échec de transmission du mail. Cela peut se traduire par exemple par le message suivant (retourné dans la boîte à lettres locale par sendmail) :

`SMTP; 552 sorry, your envelope sender domain must exist`

Dans ce cas, vous devrez ajouter une option `-f` suivie de votre adresse e-mail.

```
sendmail_path = /usr/sbin/sendmail -t -i -fmoi@monfai.com
```

Les utilisateurs de `qmail` devront probablement modifier cette valeur pour y mettre :

```
sendmail_path = /var/qmail/bin/sendmail
```


REMARQUE

Vérifier le bon fonctionnement de sendmail

Pour vérifier que sendmail fonctionne sur votre machine Linux (ou Mac OS X), il suffit d'écrire un petit fichier test en remplaçant VOTRE_ADRESSE par votre adresse :

```
Subject : Test
From : Moi <moi@moi.moi>
To : Moi <VOTRE_ADRESSE>

Cool...
```

Puis de taper `sendmail -em -t -i < test`. Vous devriez alors recevoir un mail à votre adresse.

Le fichier *php.ini* propose également le paramètre

```
mail.force_extra_parameters =
```

Les paramètres précisés ici remplaceront la valeur du cinquième paramètre de la fonction `mail()` (y compris en `safe mode`)

Envoyer un e-mail simple

La fonction permettant l'envoi d'un mail s'appelle tout simplement `mail()`. Elle peut être utilisée pour envoyer des mails tout simples ou des mails complexes (avec des fichiers joints par exemple).

mail()

Fonction permettant d'envoyer un e-mail.

Syntaxe	`boolean mail(string $vers, string $sujet, string $message[, string $entetes [, parametres]])`
`$vers`	Adresse e-mail où envoyer le courrier électronique. Vous pouvez également préciser le nom du destinataire en utilisant le format `"Nom" <email@domaine.extension>`. Si vous souhaitez mettre plusieurs personnes en destinataire, séparez simplement les adresses par une virgule.
`$sujet`	Sujet de l'e-mail.
`$message`	Le message à envoyer.
`$entetes`	En-têtes à ajouter à l'e-mail. Chaque ligne de l'en-tête doit être terminée par `"\r\n"`.
`$parametres`	Paramètres supplémentaires à passer au programme d'envoi d'e-mails (`sendmail` par exemple, introduit dans la version 4.0.5). Si `safe-mode` est activé, cet argument sera ignoré.
`retour`	Retourne `TRUE` si le message peut être envoyé (ce qui ne préjuge en rien de l'existence de l'adresse des destinataires et encore moins de la bonne réception de l'e-mail).

Une fois le serveur de courrier correctement configuré, vous pouvez tester un script PHP très simple :

Listing 12.1 : email0.php

```
<?php
    echo "Envoi email...";
    mail("testemail@toutestfacile.com", "Test d'email",
         "Ceci est un test d'email");
?>
```

Vous devriez recevoir un e-mail si vous n'oubliez pas de remplacer l'adresse testemail@toutestfacile.compar la vôtre.

Hébergeurs (gratuits)
Certains hébergeurs (notamment les gratuits) proposent une version altérée de `mail()`, souvent appelée `email()`. Le principe de fonctionnement devrait être le même, mais nous vous invitons à consulter votre fournisseur afin de connaître les différences. D'autres ne proposent tout simplement pas de fonction permettant l'envoi d'e-mails.

Comme vous êtes perspicace, vous avez pu constater que la fonction `mail()` propose quelques paramètres supplémentaires en plus des indispensables nom du destinataire, sujet et contenu du mail. Vous pouvez ainsi ajouter des informations à l'en-tête du mail, ce qui permet notamment de spécifier l'adresse de l'expéditeur, une adresse de réponse et toute une série d'informations que nous découvrirons petit à petit. Vous pouvez également passer des paramètres au client mail, ce qui est particulièrement utile pour, par exemple, préciser l'adresse à laquelle doivent être envoyés les mails d'erreur que le serveur pourrait envoyer.

Listing 12.2 : mail1.php

```
<?php
    echo "Envoi d'un email avec un entete spécifique";
    mail ("testemail@toutestfacile.com", "Test d'email",
          "Ceci est un test\nsur plusieurs lignes\n"
          ."avec un entete et un parametre",
          "From: depuistestemail@toutestfacile.com\r\n"
          ."Cc: copietestemail@toutestfacile.com\r\n"
          ."Reply-To: noreply@toutestfacile.com\r\n"
          ."X-Mailer: Mon propre mailer !",
          "-ferreurstestemail@toutestfacile.com");
?>
```

Ce script enverra donc un mail à **testemail@toutestfacile.com**, ainsi qu'une copie à **copietestemail@toutestfacile.com** (comme le précise le champ *Cc:* de l'en-tête). En répondant à ce mail, ces deux personnes écriront à **noreply@toutestfacile.com** (comme le précise le champ *Reply-To:* de l'en-tête), même si l'expéditeur sera indiqué comme étant **depuistestemail@toutestfacile.com** (comme le précise le champ *From:* de l'en-tête).

En cas d'erreur lors de l'envoi (adresse e-mail inexistante par exemple), l'e-mail d'erreur sera envoyé à **erreurstestemail@toutestfacile.com** si le programme d'envoi d'un e-mail est `sendmail` (grâce à son option *-f*).


REMARQUE

Option -f de sendmail

Si cela est possible, vous aurez tout intérêt à modifier le fichier php.ini pour préciser, au niveau du paramètre `sendmail_path`, l'option "`-f`". Cela vous évitera de répéter l'opération à chaque utilisation de la fonction `mail()`.

Voici le contenu de l'e-mail reçu (avec une partie de l'en-tête).

```
To: testemail@toutestfacile.com
Subject: Test d'email
From: depuistestemail@toutestfacile.com
CC: copietestemail@toutestfacile.com
Reply-To: noreply@toutestfacile.com
X-Mailer: Mon propre mailer !
Message-Id: <20020612005448.851652416E@toutestfacile.com>
Date: Tue, 11 Aug 2002 20:54:48 -0400 (EDT)
Content-Type: text
X-UIDL: c21"!&[M!!dAR!!(],!!

Ceci est un test
sur plusieurs lignes
avec un entete et un parametre
```


ASTUCE

Source

Pour visualiser précisément le contenu de l'en-tête du mail reçu, il vous suffit :
Avec Outlook, de sélectionner le mail et choisir l'option options du menu View (Afficher) ou bien de sauvegarder le mail et de l'ouvrir avec un éditeur de texte ;
Avec Mozilla, de sélectionner le mail et choisir l'option Message source du menu View.

Pour envoyer un même e-mail à plusieurs destinataires, il suffit de séparer les adresses par une virgule.

Type MIME

MIME (Multipurpose Internet Mail Extension) est une extension du courrier électronique ordinaire tel qu'il a été défini.

Cette norme permet de gérer des contenus qui ne sont pas du texte brut. Elle est tellement incontournable que désormais la quasi-totalité des courriers électroniques l'utilisent, même sans avoir recours à des fichiers attachés, ni même à des courriers au format HTML.

Pour qu'un message respecte la norme MIME, l'en-tête doit contenir un champ *MIME-Version* indiquant le numéro de version de la norme MIME utilisée, ainsi qu'un champ *Content-Type* précisant le type du contenu du mail et, éventuellement, d'autres champs dépendant du type sélectionné.

Le courrier précédent aurait été le suivant s'il avait été envoyé au format MIME :

```
To: testemail@toutestfacile.com
Subject: Test d'email
From: depuistestemail@toutestfacile.com
CC: copietestemail@toutestfacile.com
Reply-To: noreply@toutestfacile.com
X-Mailer: Mon propre mailer !
Message-Id: <20020612005448.851652416E@toutestfacile.com>
Date: Tue, 11 Aug 2002 20:54:48 -0400 (EDT)
MIME-Version: 1.0
Content-Type: text/plain
charset="iso-8859-1"
X-UIDL: c21"!&[M!!dAR!!(],!!

Ceci est un test
sur plusieurs lignes
avec un entete et un parametre
```

MIME
Vous trouverez une liste de RFC (en anglais) se rapportant à la norme MIME sur le site **http://www.oac.uci.edu/indiv/ehood/MIME/MIME.html**
Certaines sont traduites aux adresses :
http://jlr31130.free.fr/rfc2045-index.html
http://jlr31130.free.fr/rfc2046-index.html
http://jlr31130.free.fr/rfc2047-index.html

Envoyer un e-mail au format HTML

L'utilisation de la norme MIME permet notamment d'envoyer des fichiers au format HTML.

Pour envoyer un fichier au format HTML, il suffit de préciser que le contenu du message (*Content-type*) est de l'HTML (`text/html`). (Sinon, le destinataire verra les balises HTML qui ne seront pas interprétées.)

Listing 12.3 : mail2.php

```
<?php
    echo "Envoi d'un email HTML";
    mail("testemail@toutestfacile.com", "Test d'email",
         "<html><body>".
         "<h1>Test Email</h1>".
         "<b><u>Ceci est un document HTML</u></b><br>".
         "Avec differentes tailles de caractères et ".
         "<font color=\"red\">couleurs</font>".
         "</body></html>",
         "From: moi@monsite.com\r\n"
        ."Content-Type: text/html; charset=\"iso-8859-1\"");
 ?>
```


Envoyer un e-mail avec fichiers attachés

Vous pourrez également envoyer une image. Pour cela, il faudra bien entendu préciser le type du contenu avec *Content-type*, que nous positionnerons, dans notre cas, à la valeur `image/jpeg`. Nous pourrons également préciser qu'il s'agit d'un fichier attaché portant le nom (vu du destinataire) *image.jpg*. Pour cela, nous ajouterons à l'en-tête une ligne "`Content-Disposition: attachment; filename=image.jpg`".

Mais, surtout, nous devrons résoudre un problème. En effet, les seules données véritablement supportées par le mail sont les données de type texte. Or, nous souhaitons émettre des données binaires. Pour résoudre ce problème, il suffit de respecter les normes établies. Il faudra donc encoder le contenu binaire pour le transformer en contenu textuel (sans perte d'information). Cela peut se faire grâce à l'encodage en base 64 via la fonction `base64_encode()`.

base64_encode()

Retourne une chaîne encodée en base 64.

Syntaxe	`string base64_encode(string $chaine)`
`$chaine`	Chaîne à encoder.
`retour`	La chaîne encodée.

Afin d'indiquer au client mail le type d'encodage utilisé (ici `base64`), vous devez utiliser l'en-tête `Content-Transfert-Encoding: base64`.

Pour être totalement conforme avec la norme, les chaînes de caractères issues de l'encodage ne doivent pas dépasser 76 caractères. Il faut donc encore ajouter des retours à la ligne (`\r\n`) tous

les 76 caractères. Pour cela, vous disposez de la fonction chunk_split() qui réalise justement cette opération (cette fonction a été présentée dans le chapitre *PHP et les chaînes de caractères*).

Le script permettant l'envoi d'une image JPEG aura donc l'allure suivante :

Listing 12.4 : mail3a.php

```
<?php
    echo "Envoi d'un email avec un fichier
    ✂ attaché";
    $fichier = "imagetest.jpg";

    // Lecture du fichier en mode binaire
    (cette précision
    // n'est importante que pour windows)
    $fp = fopen($fichier, "rb");
    $fichierAttache = fread($fp, filesize($fichier));
    fclose($fp);

    // mise au format RFC
    $fichierAttache = chunk_split(base64_encode($fichierAttache));

    mail("testemail@toutestfacile.com", "Test d'email", $fichierAttache,
         "From: moi@monsite.com\r\n"
        ."Content-Disposition: attachment; filename=image.jpg\r\n"
        ."Content-Transfer-Encoding: base64\r\n"
        ."Content-Type: image/jpg\r\n");
?>
```

Figure 12.1 : *Image à envoyer (chutes du Niagara)*

Ainsi avec l'image suivante :

Ce script est équivalent à :

Listing 12.5 : mail3b.php

```
<?php
    echo "Envoi d'un email avec un fichier attaché";
    mail ("testemail@toutestfacile.com", "Test d'email",
"/9j/4AAQSkZJRgABAQEASABIAAD/2wBDAAUDBAQEAwUEBAQFBQUGBwwIBwcHBw8LCwkMEQ8SEhEP
ERETFhwXExQaFRERGCEYGhOdHx8fExciJCIeJBweHx7/wAALCABGAEYBAREA/8QAHwAAAQUBAQEB
AQEAAAAAAAAAAAECAwQFBgcICQoL/8QAtRAAAgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUEGE1Fh
ByJxFDKBkaEIIOKxwRVS0fAkM2JyggkKFhcYGRolJicoKSoONTY3ODk6QORFRkdISUpTVFVWV1hZ
WmNkZWZnaGlqc3R1dnd4eXqDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ipqrKztLW2t7i5usLDxMXG
x8jJytLT1NXW19jZ2uHi4+Tl5ufo6erx8vPO9fb3+Pn6/9oACAEBAAA/AOmmkl8+T96/3j5v96/
WX/nq/8A3aOaXgMGmWbWJ/6/30aYOs3/PWT/nq/8A3aOaYOs3/PWT/AJ2pXpXYyz/lpJ3OazvRsVpE
8f62pbVb/hvD+1/oVsUsjLqIX+Q
2SXuDjPCFh+YBqL/AIT9Qhk8Ir45x6vsnvqNn9+QCSW6eeeo56qPD+kAjzFJO1u
rEV1dlHa2ceUKAnjgcmj+OCOh4KKOhPGa1dOcsQzkhcfnVqXU4II2aVTFj7oY4Lfh1rl/EPiAzMD
Cm4xHchI43Vwd9LcSXTzu7tI5yxHFa15Av26c7f+Wjd/eojGynKFlPqGIqaGW7H3rmU+zMTV63ub
```

```
vAQSsV9M8VOOj2ZnZZLiLKAYwDgflW1e3hjg8iyijgwCCw5b8DXM3EMjzlppWYk5JJyagNpATtJU
j3ppOy2fkBPwqldIPtk3/XRv5OzyxRHFk1u6HpjTsGI+Uda6Oby4o/Kj4AqlLj86pzbQeKrTY3fd
AX196ryIOgGay7sn7bNz/wAtG/nSfMBnA5rROmza4lUBe9daojtLZYk4bHJqkz5JOeahlYnjtVRz
81NHO4HntUQAHHXHFZV5H/pc3zZO9jjHv9KdaQebKFQkk/7NdbYwLY2qsV+dhTJ5i7cmod3OKrXE
4ViqjLjseMiqkkzYZ8rsxkHHOO9R/agI2dBnCgkE4I9vrVW/nkVOZD8rLx/n8qOrnw3qBuZZFmtt
hdiAWbPX6Vp6HoFzDJvlkgbHIAyf6Vo3dhdSPkPEAOgyf8KqjS7vJ/eQ/wDfR/wpsmlXI4Lwn1yT
/hWTdaPqTTPHFLbxjdx87Hj244qGTw7qBxsnt1O7cSGPzZOcEbeaqXPh3UjKZBPbD5wWG9uuO3Hv
ULeHNY3FjeQeXOVQ7ccc9VPpX//Z",
          "From: moi@monsite.com\r\n"
         ."Content-Disposition: attachment; filename=image.jpg\r\n"
         ."Content-Transfer-Encoding: base64\r\n"
         ."Content-Type: image/jpg\r\n");
?>
```

Ce que vous pouvez facilement vérifier en sauvegardant le mail et en l'ouvrant avec un éditeur de texte.

Vous pouvez évidemment utiliser ce script pour tout type de fichier attaché (autre type d'image, exécutable, fichier son, etc.). Il vous suffit d'adapter l'en-tête `Content-type` en conséquence.

Envoyer un e-mail multi-part

Mais ne recevoir qu'un texte ou qu'une image n'a rien de palpitant. Vous serez certainement amené à accompagner votre fichier d'un message. Pour concevoir ce genre d'e-mails plus complexes, il va falloir constituer des messages électroniques avec plusieurs parties (multi-part).

Pour créer des messages avec plusieurs parties, il faut :

- Préciser dans l'en-tête principal qu'il s'agit d'un mail multi-part.
- Identifier chaque partie en la séparant des autres par un délimiteur.
- Indiquer pour chaque partie le type de contenu.

Le choix du délimiteur est important, puisqu'il ne doit pas pouvoir être confondu avec le contenu de l'e-mail, en particulier si ce mail inclut un mail multi-part (ce qui peut arriver dans le cas des mails transférés). Ce délimiteur peut être quelconque, mais il paraît évident qu'il vaut mieux choisir un délimiteur complexe et variable plutôt qu'un mot simple. Ce délimiteur est donc généralement créé aléatoirement.

Le délimiteur choisi sera indiqué au client mail en l'ajoutant à l'en-tête `Content-type`.

Voici un script commenté qui permet d'envoyer un courrier électronique contenant un texte très simple et une image en fichier attaché. Chacun de ces éléments est "stocké" dans une partie du mail. Le corps du mail, quant à lui, ne contiendra qu'un simple texte destiné aux rares clients mail ne supportant pas le format "MIME 1.0 multipart/mixed".

Listing 12.6 : mail4.php

```
<html>
<body>
Test Email avec fichier joint
<?php
```

```
//---------------------------------
// Construction de l'entête
//---------------------------------
// Le délimiteur est généré aléatoirement
$delimiteur = "-----".md5(uniqid(rand()));

// Ici, on construit un en-tête contenant les informations
// minimales requises.
// Version du format MIME utilisé
$entete = "MIME-Version: 1.0\r\n";

// Type de contenu.
// Ici plusieurs parties de type different "multipart/mixed"
// Avec un delimiteur défini par $delimiteur
$entete .= "Content-Type: multipart/mixed; boundary=\"$delimiteur\"\r\n";
$entete .= "\r\n";

//-------------------------------------------------
// Construction du message proprement dit
//-------------------------------------------------

// Pour le cas, où le logiciel de mail du destinataire
// n'est pas capable de lire le format MIME de cette version
// Il est de bon ton de l'en informer
// REM: Ce message n'apparaît pas pour les
// logiciels sachant lire ce format
$msg = "Je vous informe que ceci est un message au format MIME 1.0 ".
       "multipart/mixed.\r\n";

//---------------------------------
// 1re partie du message
// Le texte
//---------------------------------
// Chaque partie du message est séparée par une frontière
$msg .= "--$delimiteur\r\n";

// Et pour chaque partie on en indique le type
$msg .= "Content-Type: text/plain; charset=\"iso-8859-1\"\r\n";
// Et comment il sera codé
$msg .= "Content-Transfer-Encoding:8bit\r\n";
// Il est indispensable d'introduire une ligne vide
// entre l'en-tête et le texte
$msg .= "\r\n";
// Enfin, on peut écrire le texte de la 1re partie
$msg .= "Ceci est un mail avec un fichier joint\r\n";
$msg .= "\r\n";

//---------------------------------
// 2e partie du message
// Le fichier
//---------------------------------
```

```
// Tout d'abord lire le contenu du fichier
$fichier = "monfichier.jpg";
$fp      = fopen($fichier,"rb");
$fichierAttache = fread($fp, filesize($fichier));
fclose($fp);

// puis convertir le contenu du fichier en une chaîne de caractères
// certes totalement illisible, mais sans caractères exotiques
// et avec des retours à la ligne tout les 76 caractères
// pour être conforme au format RFC 2045
$fichierAttache = chunk_split(base64_encode($fichierAttache));

// Ne pas oublier que chaque partie du
// message est séparée par une frontière
$msg .= "--$delimiteur\r\n";
// Et pour chaque partie on en indique le type
$msg .= "Content-Type: image/jpg; name=\"$fichier\"\r\n";
// Et comment il sera codé
$msg .= "Content-Transfer-Encoding: base64\r\n";
// Petit plus pour les fichiers joints
// Il est possible de demander à ce que le fichier
// soit si possible affiché dans le corps du mail
$msg .= "Content-Disposition: inline; filename=\"$fichier\"\r\n";
// Il est indispensable d'introduire une ligne vide
// entre l'en-tête et le texte
$msg .= "\r\n";
// C'est ici que l'on insère le code du fichier lu
$msg .= $fichierAttache."\r\n";
$msg .= "\r\n\r\n";

// voilà, on indique la fin par une nouvelle frontière
$msg .= "--$delimiteur--\r\n";

$to    = "testemail@toutestfacile.com";
$reply = "moi@monsite.com";
$from  = "moi@monsite.com";
mail($to, "Test fichier attaché", $msg,
          "Reply-to: $reply\r\nFrom: $from\r\n".$entete);
?>
</body>
</html>
```


REMARQUE

Ligne vide

Ne pas oublier de laisser une ligne (`\r\n`) entre chaque en-tête et le message ou fichier. Il suffit de très peu de choses pour que le mail reçu ne soit pas conforme au résultat attendu.

Dans cet exemple, nous avons choisi d'afficher l'image dans le corps du mail en utilisant "`Content-Disposition: inline`". Il aurait également été possible de faire en sorte que l'image soit jointe au mail en tant que fichier à sauvegarder par l'utilisateur, en remplaçant

inline par attachment. La nuance entre les deux dépendra également du client mail utilisé par le destinataire (en effet, certains ne supportent pas le mode inline).

Le script suivant comporte deux fichiers attachés un en '`inline`' et un en '`attachment`'. Vous serez ainsi en mesure de voir la différence.

Listing 12.7 : mail5.php

```
<html>
<body>
Test Email avec 2 fichiers joints
<?php

//---------------------------------
// Construction de l'entête
//---------------------------------
$delimiteur = "-----=".md5(uniqid(rand()));

$entete = "MIME-Version: 1.0\r\n";
$entete .= "Content-Type: multipart/mixed; boundary=\"$delimiteur\"\r\n";
$entete .= "\r\n";

//-----------------------------------------------------
// Construction du message proprement dit
//-----------------------------------------------------

$msg = "Je vous informe que ceci est un message au format MIME 1.0".
       " multipart/mixed.\r\n";

//---------------------------------
// 1re partie du message
// Le texte
//---------------------------------
$msg .= "--$delimiteur\r\n";
$msg .= "Content-Type: text/plain; charset=\"iso-8859-1\"\r\n";
$msg .= "Content-Transfer-Encoding:8bit\r\n";
$msg .= "\r\n";
$msg .= "Ceci est un mail avec 2 fichiers joints\r\n";
$msg .= "\r\n";

//---------------------------------
// 2e partie du message
// Le 1er fichier (inline)
//---------------------------------
$fichier = "monfichier.jpg";
$fp = fopen($fichier,"rb");
$fichierAttache = fread($fp, filesize($fichier));
fclose($fp);
$fichierAttache=chunk_split(base64_encode($fichierAttache));

$msg .= "--$delimiteur\r\n";
$msg .= "Content-Type: image/jpg; name=\"$fichier\"\r\n";
```

```
$msg .= "Content-Transfer-Encoding: base64\r\n";
$msg .= "Content-Disposition: inline; filename=\"$fichier\"\r\n";
$msg .= "\r\n";
$msg .= $fichierAttache . "\r\n";
$msg .= "\r\n\r\n";

//---------------------------------
// 3ème partie du message
// Le 2ème fichier (attachment)
//---------------------------------
$fichier = "monfichier2.jpg";
$fp      = fopen($fichier,"r");
$fichierAttache = fread($fp,filesize($fichier));
fclose($fp);
$fichierAttache = chunk_split(base64_encode($fichierAttache));

$msg .= "--$delimiteur\r\n";
$msg .= "Content-Type: image/jpg; name=\"$fichier\"\r\n";
$msg .= "Content-Transfer-Encoding: base64\r\n";
$msg .= "Content-Disposition: attachment; filename=\"$fichier\"\r\n";
$msg .= "\r\n";
$msg .= $fichierAttache . "\r\n";
$msg .= "\r\n\r\n";

$msg .= "--$delimiteur--\r\n";

$to    = "testemail@toutestfacile.com";
$reply = "moi@monsite.com";
$from  = "moi@monsite.com";
mail($to, "Test avec 2 fichiers attachés", $msg,
     "Reply-to: $reply\r\nFrom: $from\r\n".$entete);
?>
</body>
</html>
```


REMARQUE

Client e-mail

Vous pouvez ne pas voir de différence selon votre client e-mail.

Envoyer un e-mail HTML avec des images

Envoyer un courrier électronique au format HTML incluant des images est à peine plus compliqué. Il existe en fait deux cas de figure. Le plus simple, mais le moins satisfaisant, consiste à considérer que les images sont disponibles sur un serveur web. Il suffit, dans ce cas, de reprendre le premier exemple d'envoi d'un e-mail au format HTML, et de préciser des adresses complètes pour les images (du genre `<img src="http://monsite.com/monimage.jpg">`). L'autre solution consiste à envoyer les images avec le corps du mail HTML. Pour cela, il suffit de créer un mail multi-part, comme dans les exemples précédents. Mais la difficulté (si l'on peut dire) apparaît lorsqu'il s'agit de faire référence à ces fichiers attachés depuis les balises HTML.

Pour répondre à ce besoin, il est possible de donner un identifiant à une partie quelconque du mail (ici, les images), grâce à l'en-tête `Content-ID:` suivi d'une chaîne de caractères choisie comme identifiant, et placée entre '<' et '>'. Pour faire ensuite appel à ces éléments, il suffit de préciser l'identifiant de la partie précédé de "`cid:`".

Ce qui donne un script du genre :

Listing 12.8 : mail6.php

```
<html>
<body>
Test Email au format HTML avec 2 images
<?php

//---------------------------------
// Construction de l'en-tête
//---------------------------------
$delimiteur = "-----=".md5(uniqid(rand()));

$entete = "MIME-Version: 1.0\r\n";
$entete .= "Content-Type: multipart/mixed; boundary=\"$delimiteur\"\r\n";
$entete .= "\r\n";

//-----------------------------------------------------
// Construction du message proprement dit
//-----------------------------------------------------

$msg = "Je vous informe que ceci est un message au format MIME 1.0".
       " multipart/mixed.\r\n";

//---------------------------------
// 1ère partie du message
// Le code HTML
//---------------------------------
$msg .= "--$delimiteur\r\n";
$msg .= "Content-Type: text/html; charset=\"iso-8859-1\"\r\n";
$msg .= "Content-Transfer-Encoding:8bit\r\n";
$msg .= "\r\n";
$msg .= "<html><body><h1>Email HTML avec 2 images</h1>";
$msg .= "<table>";
$msg .= "<tr><th>Image 1</th></tr><tr><td><img src=\"cid:image1\"></td></tr>";
$msg .= "<tr><th>Image 2</th></tr><tr><td><img src=\"cid:iamge2\"></td></tr>";
$msg .= "</table>";
$msg. = "</body></html>\r\n";
$msg.="\r\n";

//---------------------------------
// 2e partie du message
// Le 1er fichier
//---------------------------------
$fichier = "monfichier.jpg";
$fp      = fopen($fichier, "rb");
$fichierAttache = fread($fp, filesize($fichier));
```

```
fclose($fp);
$fichierAttache = chunk_split(base64_encode($fichierAttache));

$msg .= "--$delimiteur\r\n";
$msg .= "Content-Type: application/octet-stream; name=\"$fichier\"\r\n";
$msg .= "Content-Transfer-Encoding: base64\r\n";
$msg .= "Content-ID: <image1>\r\n";
$msg .= "\r\n";
$msg .= $fichierAttache . "\r\n";
$msg .= "\r\n\r\n";

//---------------------------------
// 3ème partie du message
// Le 2ème fichier
//---------------------------------
$fichier = "monfichier2.jpg";
$fp = fopen($fichier, "rb");
$fichierAttache = fread($fp, filesize($fichier));
fclose($fp);
$fichierAttache = chunk_split(base64_encode($fichierAttache));

$msg .= "--$delimiteur\r\n";
$msg .= "Content-Type: application/octet-stream; name=\"$fichier\"\r\n";
$msg .= "Content-Transfer-Encoding: base64\r\n";
$msg .= "Content-ID: <image2>\r\n";
$msg .= "\r\n";
$msg .= $fichierAttache . "\r\n";
$msg .= "\r\n\r\n";

$msg .= "--$delimiteur--\r\n";

$to    = "testemail@toutestfacile.com";
$reply = "moi@monsite.com";
$from  = "moi@monsite.com";
mail($to, "Email HTML avec 2 images", $msg,
          "Reply-to: $reply\r\nFrom: $from\r\n".$entete);

echo "A destination de $to<br />";
?>
</body>
</html>
```

Envoyer un courrier électronique en PHP est d'une simplicité déconcertante, non ?

12.2. Accéder à son compte messagerie IMAP, POP3 ou NNTP

Accéder à un compte IMAP, POP3, ou NNTP (serveur de 'news') permet de consulter des messages électroniques depuis n'importe quel navigateur, sans avoir à le reconfigurer. Cela

permet aussi de personnaliser l'interface et, ainsi, d'autoriser ou d'interdire certaines fonctions par rapport à un client email "classique". Ce principe d'accès aux messages électroniques via un site Internet est communément appelé *webmail*.

Que ce soit pour accéder à un compte IMAP, POP3 ou NNTP, le principe est presque identique, à quelques petites différences près.

Avant toute chose, il faut installer la bibliothèque IMAP.

Installation

Sous Windows

Avec l'archive du PHP Group

Vous devez, dans un premier temps, vous assurer de disposer du fichier *php_imap.dll* (fourni avec l'archive) dans le répertoire des extensions PHP. Il vous suffit alors de modifier le fichier *php.ini* pour ajouter ou décommenter une ligne.

```
extension=php_imap.dll
```

Avec EasyPHP

Avec EasyPHP, le support d'IMAP est activé par défaut.

Sous Linux

Vous devez, dans un premier temps, récupérer la bibliothèque cliente IMAP (*c-client.tar.Z*) sur le site ftp://ftp.cac.washington.edu/imap/ (également disponible sur le CD-ROM fourni).

Vous pouvez la copier dans un répertoire quelconque (par exemple, */usr/local/src/lib*) , puis décompressez l'archive avec :

```
# uncompress c-client.tar.Z
# tar xvf c-client.tar
```

Puis compilez-la :

```
# cd imap-2001a
# make slx
```

Autres systèmes d'exploitation
Vous aurez peut-être à choisir une autre option si vous souhaitez adapter cette procédure d'installation pour un autre système d'exploitation (les options sont décrites dans le fichier Makefile).

Vous devez avoir ainsi généré un ensemble de fichiers sous le répertoire *c-client*. Certains de ces fichiers doivent alors être copiés dans des répertoires accessibles lors de la compilation de PHP.

```
# cp c-client/c-client.a /usr/local/lib/.
```

et

```
# cp c-client/rfc822.h /usr/local/include/.
# cp c-client/mail.h /usr/local/include/.
# cp c-client/linkage.h /usr/local/include/.
```

Une fois cette opération réalisée, vous pouvez recompiler PHP avec l'option `--with-imap`.

Vous pouvez vous reporter au chapitre "Prise en main" pour plus de détails sur la compilation de PHP.

Vérification

Vous pouvez vérifier que votre environnement PHP intègre bien le support d'`IMAP` en appelant un simple script contenant `<?php phpinfo(); ?>` qui devra afficher :

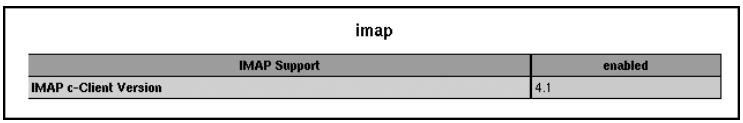


Figure 12.2 : *phpinfo()*

Connexion et déconnexion à un serveur

Pour pouvoir interroger un compte, il faut commencer par se connecter à celui-ci. Pour cela, vous devez préalablement réunir quelques informations.

Il faut connaître :

- L'adresse du serveur de mail (adresse IP ou nom complet) ;
- Le port du service (généralement 110 pour POP3, 143 pour IMAP et 119 pour NNTP) ;
- Le login d'accès (celui qui sert à accéder au compte habituellement) ;
- Le mot de passe associé au login précédent.

Dans nos exemples, la connexion se fera sur **mail.monsite.com** avec le login `monlogin` et le mot de passe `monpassword`.

Pour les opérations de connexion, les boîtes à lettres sont représentées par une chaîne de caractères ayant le format : `{<serveur>:<port>/<type>}<boite>`.

Autrement dit, elle se compose de deux parties :

- Une partie entre accolades indiquant le serveur. On y retrouve, l'adresse IP ou le nom du serveur, puis le numéro du port précédé de `:`. Puis enfin le type de serveur (IMAP par défaut) en ajoutant `/imap`, `/pop` ou `/nntp`. Dans le cas d'un serveur sécurisé, il faut ajouter `/ssl` après le type de serveur et, si le serveur est sécurisé avec un certificat, il faut ajouter `/ssl/novalidate-cert`.
- Puis le nom de la boîte à lettres à ouvrir (INBOX par défaut).

Nom de boîte et caractères spéciaux
Si le nom de la boîte à lettres doit contenir des caractères internationaux peu courants, alors il doit être encodé avec la fonction `imap_utf7_encode()` *avant d'être utilisé comme référence ou décodé avec* `imap_utf7_decode()` *lorsqu'il est lu.*

imap_open()

Permet d'ouvrir une connexion sur un serveur IMAP, POP3 ou NNTP.

Syntaxe	`resource imap_open(string $bal, string $login, string $motDePasse [, int $mode])`
`$bal`	Identifiant de la boîte à lettres au format `{<serveur>:<port>/<type>}<boite>`.
`$login`	Login de connexion.
`$motDePasse`	Mot de passe correspondant au login.
`$mode`	Une combinaison par `OU` logique des valeurs. `OP_ANONYMOUS` : n'utilise pas de `.newsrc` (uniquement pour NNTP). `OP_HALFOPEN` : permet d'ouvrir une connexion sans sélectionner de boîte (uniquement pour IMAP et NNTP). `OP_READONLY` : permet d'ouvrir une boîte en lecture seule. `CL_EXPUNGE` : efface automatiquement les messages marqués comme étant à effacer de la boîte lors de la fermeture.
`retour`	Un identifiant, ou `FALSE` si la connexion n'a pas pu se faire.

Voici quelques exemples :

Pour ouvrir la boîte à lettres INBOX d'un serveur IMAP sur le port 143 situé sur la machine courante :

```
imap_open("{localhost:143}INBOX", "monlogin", "mon password")
```

Pour ouvrir la boîte à lettres OUTBOX en lecture seule d'un serveur POP3 sur le port 110 situé sur la machine *mail.monsite.com* :

```
imap_open("{mail.monsite.com:110/pop3}OUTBOX", "monlogin", "mon password",
          "OP_READONLY")
```

Pour ouvrir le groupe `comp.lang.php` du serveur de news NNTP sur le port 119 situé sur la machine *news.monfai.com* :

```
imap_open("{news.monfai.com:119/nntp}comp.lang.php", "", "")
```

imap_close()

Permet de fermer une connexion à un serveur POP3, IMAP ou NNTP.

Syntaxe	`boolean imap_close(resource $identifiantBal [, int $mode])`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$mode`	Le seul mode disponible est : `CL_EXPUNGE` qui fait effacer tous les messages marqués comme étant à effacer de la boîte à lettres en quittant.
`retour`	`TRUE` si l'opération s'est bien effectuée, `FALSE` dans le cas contraire.

Un script de base de connexion et de déconnexion a donc l'allure suivante :

Listing 12.9 : imap_1.php

```
<html>
<head><title>Exemple IMAP</title></head>
<body>
<?php
    $bal= imap_open("{mail.monsite.com:110/pop3}",
                    "monlogin",
                    "monpassword");
    if (!$bal) die("La connexion a échouée");
    imap_close($bal);
?>
</body>
</html>
```

Il est également possible à tout instant de vérifier la validité d'une connexion.

imap_ping()

Vérifie que la connexion IMAP est toujours active.

Syntaxe	`boolean imap_ping(resource $identifiantBal)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`retour`	`TRUE` si la connexion est toujours active.

Une fois que vous êtes connecté, de nombreuses possibilités s'offrent à vous. Cela va de la récupération de la liste des boîtes à lettres à leur administration, en passant par la lecture de leur contenu.

Sélection d'une boîte à lettres

Si vous vous êtes connecté à un serveur sans ouvrir de boîte à lettres (mode `OP_HALFOPEN`) ou, tout simplement, si vous souhaitez accéder à une autre boîte à lettres, vous disposez de la fonction :

imap_reopen()

Ouvre une nouvelle connexion IMAP vers une nouvelle boîte à lettres.

Syntaxe	`boolean imap_reopen(resource $identifiantBal, string $bal [, int $mode])`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$bal`	Nouvelle boîte à lettres à ouvrir.
`$mode`	Une combinaison par `OU` logique des valeurs. `OP_ANONYMOUS` : n'utilise pas de `.newsrc` (uniquement pour NNTP). `OP_HALFOPEN` : permet d'ouvrir une connexion sans sélectionner de boîte (uniquement pour IMAP et NNTP). `OP_READONLY` : permet d'ouvrir une boîte en lecture seule. `CL_EXPUNGE` : efface automatiquement les messages marqués comme étant à effacer de la boîte lors de la fermeture.
`retour`	Un identifiant, ou `FALSE` si la connexion n'a pas pu se faire.

Pour vous aider à faire votre choix, vous aurez peut-être besoin de connaître la liste des boîtes disponibles. Vous disposez pour cela de plusieurs fonctions.

imap_listMailbox()

Retourne la liste des boîtes à lettres.

Syntaxe	`array imap_listMailbox(resource $identifiantBal, string $serveur, string $position)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$serveur`	Serveur au format `{<serveur>:<port>/<type>}`.
`$position`	Position hiérarchique de la première boîte à étudier. Vous pouvez utiliser le caractère `*` pour tout étudier ou `%` pour toutes les boîtes du niveau sélectionné (ex : `comp.lang.%`).
`retour`	Tableau indexé des noms complets des boîtes à lettres.

Voici un script d'exemple :

Listing 12.10 : imap_listmailbox.php

```
<?php
$bal = imap_open("{mail.monsite.com:143/imap}","monlogin","monpassword");
$tableau = imap_listmailbox($bal, "{mail.monsite.com}", "*");
while (list($cle, $valeur) = each($tableau)) {
    echo "$cle:";
    echo imap_utf7_decode($valeur)."<br />\n";
}
imap_close($bal);
?>
```

Et un résultat possible :

```
0:{mail.monsite.com}INBOX.Sent.mai-2002
1:{mail.monsite.com}INBOX.Trash
2:{mail.monsite.com}INBOX.Sent
3:{mail.monsite.com}INBOX.Drafts
4:{mail.monsite.com}INBOX
```

Mais peut-être voudrez vous restreindre votre recherche.

imap_scanmailbox()

Recherche une chaîne de caractères dans les différentes boîtes à lettres et retourne les noms de celles-ci.

Syntaxe	`array imap_scanmailbox(resource $identifiantBal, string $serveur, string $position, string $chaineARechercher)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$serveur`	Serveur au format `{<serveur>:<port>/<type>}`.
`$position`	Position hiérarchique de la première boîte à étudier. Vous pouvez utiliser le caractère `*` pour tout étudier ou `%` pour toutes les boîtes du niveau sélectionné (ex. : `comp.lang.%`).
`$chaineARechercher`	Chaîne de caractères à rechercher dans les différentes boîtes à lettres.
`retour`	Un tableau des différentes boîtes à lettres.

Si, en revanche, vous souhaitez en savoir un peu plus sur les boîtes à lettres, en théorie vous disposez de `imap_getMailboxes()` (ne nous sommes toutefois pas parvenus à obtenir le résultat escompté).

imap_getMailboxes()

Retourne la liste des boîtes à lettres, en précisant si elles possèdent des boîtes "filles" ou non, ainsi que le délimiteur hiérarchique.

Syntaxe	`array imap_getMailboxes(resource $identifiantBal, string $serveur, string $position)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$serveur`	Serveur au format `{<serveur>:<port>/<type>}`.
`$position`	Position hiérarchique de la première boîte à étudier. Vous pouvez utiliser le caractère `*` pour tout étudier ou `%` pour toutes les boîtes du niveau sélectionné (ex. : `comp.lang.%`).
`retour`	Tableau indexé d'objets possédant les attributs : "`name`": nom complet de la boîte aux lettres. "`delimiter`" : délimiteur hiérarchique (i.e. caractère à utiliser pour accéder à une boîte fille. Il s'agit généralement du point `boitemere.boitefille`). "`attributes`" : peut prendre les valeurs `LATT_NOINFERIORS` s'il n'y a pas de boîte à lettres en dessous de celle-ci, `LATT_NOSELECT` si celle-ci n'est qu'un conteneur, `LATT_MARKED` si elle est marquée, `LATT_UNMARKED` si elle n'est pas marquée.

Voici un exemple de script :

Listing 12.11 : imap_getmailboxes.php

```
<?php
$bal = imap_open("{mail.monsite.com:143/imap}","monlogin","monpassword");
$tableau = imap_getMailboxes($bal, "{mail.monsite.com}", "*");
while (list($cle, $valeur) = each($tableau)) {
    echo "$cle:";
    echo imap_utf7_decode($valeur->name).",";
    echo "'".$valeur->delimiter."',";
    echo $valeur->attributes."<br>\n";
}
imap_close($bal);
?>
```

dont un des résultat pourrait être :

```
0:{mail.monsite.com}INBOX.Sent.mai-2002,'.',0
1:{mail.monsite.com}INBOX.Trash,'.',0
2:{mail.monsite.com}INBOX.Sent,'.',0
3:{mail.monsite.com}INBOX.Drafts,'.',0
4:{mail.monsite.com}INBOX,'.',4
```

Si vous souhaitez limiter la liste aux boîtes ou (plus probablement) aux news auxquelles vous avez souscrit, alors vous disposez des fonctions `imap_listSubscribed()` et `imap_getSubscribed()`.

imap_listSubscribed()

Récupère les boîtes auxquelles l'utilisateur a souscrit. L'utilisation est en tous points identique à `imap_listMailboxes()`.

Syntaxe	`array imap_listSubscribed(resource $identifiantBal, string $serveur, string $position)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$serveur`	Serveur au format `{<serveur>:<port>/<type>}`.
`$position`	Position hiérarchique de la première boîte à étudier. Vous pouvez utiliser le caractère `*` pour tout étudier ou `%` pour toutes les boîtes du niveau sélectionné (ex. : `comp.lang.%`).
`retour`	Tableau indexé des noms complets des boîtes.

Si, en revanche, vous souhaitez en savoir un peu plus sur les boîtes à lettres...

imap_getSubscribed()

Récupère les boîtes auxquelles l'utilisateur a souscrit. L'utilisation est en tous points identique à `imap_getMailboxes()`.

Syntaxe	`array imap_getSubscribed(resource $identifiantBal, string $serveur, string $position)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$serveur`	Serveur au format `{<serveur>:<port>/<type>}`.
`$position`	Position hiérarchique de la première boîte à étudier. Vous pouvez utiliser le caractère `*` pour tout étudier ou `%` pour toutes les boîtes du niveau sélectionné (ex. : `comp.lang.%`).
`retour`	Tableau indexé d'objets possédant les attributs : "`name`" : nom complet de la boîte à lettres. "`delimiter`" : délimiteur hiérarchique (i.e. caractère à utiliser pour accéder à une boîte fille. Il s'agit généralement du point `boitemere.boitefille`). "`attributes`" : peut prendre les valeurs `LATT_NOINFERIORS` s'il n'y a pas de boîte à lettres en dessous de celle-ci, `LATT_NOSELECT` si celle-ci n'est qu'un conteneur, `LATT_MARKED` si elle est marquée, `LATT_UNMARKED` si elle n'est pas marquée.

Aperçu du contenu de la boîte à lettres

imap_check()

Retour des informations sur la boîte à lettres courante.

Syntaxe	`object imap_check(resource $identifiantBal)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`retour`	Objet contenant les attributs suivants : "`Date`" : Date du serveur. "`Driver`" : Protocole utilisé pour accéder à la boîte e-mail. POP3, IMAP ou NNTP. "`Mailbox`" : Nom complet de la boîte de courrier électronique. "`Nmsgs`" : Nombre de messages dans la boîte. "`Recent`" : Nombre de messages marqués 'récent' dans la boîte.

Listing 12.12 : imap_check.php

```
<?php
    $bal    = imap_open("{mail.monsite.com:143/imap}INBOX",
                        "monlogin", "monpassword");
    $objet = imap_check($bal);
    echo $objet->Date."<br />\n";
    echo $objet->Driver."<br />\n";
    echo $objet->Mailbox."<br />\n";
    echo $objet->Nmsgs."<br />\n";
    echo $objet->Recent."<br />\n";
    imap_close($bal);
?>
```

dont voici un exemple de résultat :

```
Mon, 17 Jun 2002 21:18:06 -0400 (Est (heure d'été))
imap
{ns0.unsite.net:143/imap/user="monlogin "}INBOX
11
0
```

Il est toutefois possible d'en savoir un peu plus, comme le démontre la fonction suivante.

imap_mailboxMsgInfo()

Retourne des informations sur la boîte à lettres courante.

Syntaxe	`object imap_mailboxmsginfo(resource $identifiantBal)`
`$identifiantBal`	Identifiant retourné par `imap_open()`.
retour	Un objet contenant les attributs : "`Date`" : Date du dernier changement. "`Driver`" : Driver. "`Mailbox`" : Nom complet de la boîte aux lettres. "`Nmsgs`" : Nombre de messages. "`Recent`" : Nombre de messages marqués Recent. "`Unread`" : Nombre de messages marqués Unread (non lus). "`Deleted`" : Nombre de messages marqués Deleted (effacés). "`Size`" : Taille de la boîte aux lettres.

Voici un script d'exemple :

Listing 12.13 : imap_mailboxmsginfo.php

```
<?php
$bal = imap_open("{mail.monsite.com:143/imap}",
       "monlogin", "monpassword");
$objet = imap_mailboxmsginfo($bal);
echo "Date: " . $objet->Date ."<br />\n" ;
echo "Driver: " . $objet->Driver ."<br />\n" ;
echo "Mailbox: " . $objet->Mailbox ."<br />\n" ;
echo "Messages: ". $objet->Nmsgs ."<br />\n" ;
echo "Recent: " . $objet->Recent ."<br />\n" ;
echo "Unread: " . $objet->Unread ."<br />\n" ;
echo "Deleted: " . $objet->Deleted ."<br />\n" ;
echo "Size: " . $objet->Size ."<br />\n" ;
imap_close($bal);
?>
```

dont le résultat serait par exemple :

```
Date: Wed, 19 Jun 2002 20:45:04 -0400 (Est (heure d'été))
Driver: imap
Mailbox: {ns0.monsite.net:143/imap/user="monlogin"}INBOX
Messages: 3
Recent: 0
Unread: 3
Deleted: 0
Size: 1832
```

Vous disposez également de deux fonctions récupérant spécifiquement le nombre de messages et le nombre de messages récents.

imap_num_msg()

Renvoie le nombre de messages de la boîte à lettres courante.

Syntaxe	`int imap_num_msg(resource $identifiantBal)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`retour`	Nombre de messages.

imap_num_recent()

Renvoie le nombre de messages marqués "Recent" de la boîte à lettres courante.

Syntaxe	`int imap_num_recent(resource $identifiantBal)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`retour`	Nombre de messages marqués "Recent".

Mais il est également possible de profiter de la connexion ouverte sur le serveur pour glaner quelques informations sur d'autres boîtes à lettres.

imap_status()

Cette fonction retourne des informations sur une boîte à lettres différente de la courante.

Syntaxe	`object imap_status(resource $identifiantBal, string $bal, int $options)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$bal`	Boîte à lettres sur laquelle vous souhaitez des informations.
`$options`	Combinaison par `OU` logique des valeurs suivantes permettant de déterminer quelles informations doivent être retournées (sous forme d'attributs d'un objet) : `SA_MESSAGES` (= 1) pour stocker le nombre de messages dans l'attribut "`message`". `SA_RECENT` (= 2) pour stocker le nombre de messages récents dans l'attribut "`recent`" `SA_UNSEEN` (= 4) pour stocker le nombre de messages non lus dans l'attribut "`unseen`" `SA_UIDNEXT` (= 8) pour stocker le prochain UID qui sera utilisé pour la boîte à lettres dans l'attribut "`uidnext`". `SA_UIDVALIDITY` (= 16) pour stocker dans l'attribut "`uidvalidity`" une chaîne de caractères qui change quand la boîte à lettres peut ne plus être valide. d`SA_ALL` (= 31) pour attribuer toutes les valeurs précédentes.

`retour`	Un objet avec les attributs demandés ainsi que l'attribut `"flags"` contenant la valeur de `$options`.

Lecture des en-têtes

Il est fort heureusement possible de récupérer la liste des en-têtes des messages contenus dans une boîte via la fonction `imap_headers()`.

imap_headers()

Permet de retourner les résumés d'en-têtes de tous les messages d'une boîte à lettres.

Syntaxe	`array imap_headers(resource $identifiantBal)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`retour`	Tableau associatif contenant tous les en-têtes (une simple chaîne résumée par message). Un tableau vide s'il n'y a pas de message.

Ce qui nous permet d'écrire un script qui commence à être intéressant :

Listing 12.14 : imap_2.php

```
<html>
<head><title>Exemple IMAP</title></head>
<body>
<p><h1>Entetes de mail dans INBOX</h1>
<?php
$mbox = imap_open("{mail.monsite.com:110/pop3}", "monlogin", "monpassword");
$headers = imap_headers($mbox);
if (!$headers) {
    echo "Erreur !\n";
} else {
    while (list($key,$val) = each($headers)) {
        echo $val."<br>\n";
    }
}
imap_close($mbox);
?>
</body>
</html>
```

dont le résultat pourrait être le suivant :

Figure 12.3 :
Lister des messages

Il est également possible d'avoir quelques informations supplémentaires sur l'ensemble ou un sous-ensemble de messages.

imap_fetch_overview()

Donne un aperçu de l'en-tête d'un des messages.

Syntaxe	`array imap_fetch_overview(resource $identifiantBal, string $sequence [, int $mode])`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$listeMsg`	Liste des numéros (ou UID) ou intervalles de numéros (ou UID) de messages séparés par une virgule (le premier porte l'indice 1). Les intervalles sont définis par le numéro de début, le caractère `:` et le numéro de fin.
`$mode`	`FT_UID` si le numéro du message est son UID.
`retour`	Un tableau indexé d'objets contenant les attributs suivants :

- "`subject`" : le sujet du message.
- "`from`" : l'auteur du message.
- "`date`" : la date de l'envoi du message.
- "`message_id`" : l'identifiant du message.
- "`references`" : référence à l'identifiant.
- "`size`" : taille du mail en octets.
- "`uid`" : UID du message dans la boîte aux lettres.
- "`msgno`" : numéro du message dans la boîte.
- "`recent`" : indique si le message est marqué récent.
- "`flagged`" : indique si le message est marqué.
- "`answered`" : indique si le message a été répondu.

- "deleted" : indique si le message est marqué comme étant à effacer.
- "seen" : indique si le message a été marqué lu.
- "draft" : Indique si le message a été marqué comme brouillon.

Voici un petit exemple :

Listing 12.15 : imap_fetch_overview.php

```
<?php
    $bal = imap_open("{mail.monsite.com:143/imap}INBOX",
                     "monlogin", "monpassword");
    $tableau = imap_fetch_overview($bal, "1,3");
    foreach ($tableau as $element) {
        echo $element->from."<br />\n";
        echo $element->date."<br />\n";
    }
    imap_close($bal);
?>
```

Et son résultat pourrait être :

```
Damien
Sat, 26 Jan 2002 00:19:48 +0100
Thomas HEUTE
Tue, 18 Jun 2002 17:49:08 -0400
```

Il est toutefois possible d'avoir une connaissance plus détaillée des en-têtes grâce, en particulier, aux fonctions `imap_fetchHeader()` et `imap_headerInfo()`.

imap_fetchHeader()

Retourne l'en-tête d'un message.

Syntaxe	`string imap_fetchHeader(resource $identifiantBal, int $numeroMsg [, int $mode])`
`$identifiantBal`	Identifiant retourné par `imap_open()`.
`$numeroMsg`	Indice du message à ouvrir (le premier porte l'indice 1).
`$mode`	Combinaison par `OU` logique des options : *FT_UID* indiquant que le numéro du message précisé est son UID. *FT_INTERNAL* pour retourner l'en-tête dans le format 'interne'. *FT_PREFETCHTEXT*.
`retour`	L'en-tête du message désiré.

Afin d'en simplifier l'analyse, vous disposez d'une fonction transformant cette chaîne de caractères en un objet où chaque attribut est un élément de l'en-tête.

imap_rfc822_parse_headers()

Permet d'extraire des informations d'un en-tête.

Syntaxe	`object imap_rfc822_parse_headers(string $entete [, string $hoteDefaut])`
`$entete`	L'en-tête d'un courrier électronique tel que retourné par `imap_fetchHeader()`.
`$hoteDefaut`	Le nom de domaine par défaut.
`retour`	Un objet contenant de nombreux attributs (dont la liste est ci-après) correspondant aux différents champs que vous pouvez rencontrer dans l'en-tête du message.

- `remail`: redirection automatique de mails.
- `date`, `Date` : date du message.
- `subject`, Subject : sujet du message.
- `in_reply_to` : adresse à laquelle le message répond.
- `message_id` : identifiant du message.
- `newsgroups`.
- `followup_to` : adresse de suivi du message.
- `references`.
- `toaddress` : ligne d'en-tête de '`TO:`' limitée à 1024 caractères.
- `to[]` : tableau de toutes les adresses de '`TO:`' sous forme d'objet.
- `to[]->personal` : nom de la personne.
- `to[]->adl` : at domain source route (???).
- `to[]->mailbox` : partie précédant '`@`'.
- `to[]->host` : partie suivant '`@`'.
- `fromaddress` : ligne d'en-tête de '`FROM:`' limitée à 1024 caractères.
- `from[]`: tableau de toutes les adresses de '`FROM:`' sous forme d'objet.
- `from[]->personal` : nom de la personne.
- `from[]->adl` : at domain source route.
- `from[]->mailbox` : partie précédant '`@`'.
- `from[]->host`: partie suivant '`@`'.
- `ccaddress` : ligne d'en-tête de '`CC:`' limitée à 1024 caractères.
- `cc[]` : tableau de toutes les adresses de '`CC:`' sous forme d'objet.
- `cc[]->personal` : nom de la personne.
- `cc[]->adl` : at domain source route.
- `cc[]->mailbox` : partie précédant '`@`'.

- `cc[]->host` : partie suivant '`@`'.
- `bccaddress` : ligne d'en-tête de '`BCC:`' limitée à 1024 caractères.
- `bcc[]` : tableau de toutes les adresses de '`BCC:`' sous forme d'objet.
- `bcc[]->personal` : nom de la personne.
- `bcc[]->adl`: at domain source route.
- `bcc[]->mailbox` : partie précédant '`@`'.
- `bcc[]->host` : partie suivant '`@`'.
- `reply_toaddress` : ligne d'en-tête de '`Reply_to:`' limitée à 1024 caractères.
- `reply_to[]` : tableau de toutes les adresses de '`Reply_to:`' sous forme d'objet.
- `reply_to[]->personal` : nom de la personne.
- `reply_to[]->adl` : at domain source route.
- `reply_to[]->mailbox` : partie précédant '`@`'.
- `reply_to[]->host` : partie suivant '`@`'.
- `senderaddress` : ligne d'en-tête de '`Sender:`' limitée à 1024 caractères.
- `sender[]` : tableau de toutes les adresses de '`Sender:`' sous forme d'objet.
- `sender[]->personal` : nom de la personne.
- `sender[]->adl` : at domain source route.
- `sender[]->mailbox` : partie précédant '`@`'.
- `sender[]->host` : partie suivant '`@`'.
- `return_path[]` : tableau de toutes les adresses de '`Return-path:`' sous forme d'objet.
- `return_path[]->personal` : nom de la personne.
- `return_path[]->adl` : at domain source route.
- `return_path[]->mailbox` : partie précédant '`@`'.
- `return_path[]->host` : partie suivant '`@`'.

Vous pouvez toutefois vous dispenser de faire deux opérations tout en récupérant plus d'informations (on se demande bien comment c'est possible).

imap_headerInfo()

Permet de lire l'en-tête d'un message.

Syntaxe `object imap_headerInfo(resource $identifiantBal, int $numeroMesg [, int $tailleFrom [, int $tailleSubject [, string hoteDefaut]]])`

`$identifiantBal` Identifiant tel que retourné par `imap_open()`.

`$numeroMesg` Position du message à ouvrir (le premier porte l'indice 1).

`$tailleFrom`	Taille maximale de la chaîne de caractères `From`.
`$tailleSubject`	Taille maximale de la chaîne de caractères `Subject`.
`$hoteDefaut`	Hôte par défaut.
`retour`	Un objet contenant de nombreux attributs (dont la liste est ci-après) correspondant aux différents champs que vous pouvez rencontrer dans l'en-tête du message.

`imap_headerInfo()` possède un alias appelé `imap_header()` (à éviter à cause du risque de confusion avec `imap_headers()`).

En voici quelques-unes récupérées du courrier électronique suivant :

```
Return-Path: <webmaster@toutestfacile.com>
Delivered-To: copiecacheebcc@toutestfacile.com
Received: (qmail 15229 invoked by uid 503); 17 Jun 2002 02:05:14 -0000
Received: from unknown (HELO thomas) (138.88.143.137)
  by ns0.ovh.net with SMTP; 17 Jun 2002 02:05:14 -0000
Date: Sun, 16 Jun 2002 22:04:14 -0400
From: ToutEstFacile <webmaster@toutestfacile.com>
X-Mailer: The Bat! (v1.60m) UNREG / CD5BF9353B3B7091
Reply-To: ToutEstFacile <webmaster@toutestfacile.com>
Organization: ToutEstFacile
X-Priority: 3 (Normal)
Message-ID: <10913313113.20020616220414@toutestfacile.com>
To: imap@toutestfacile.com, copie@toutestfacile.com
CC: copieCC@toutestfacile.com
Subject: Ceci est le sujet du mail
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

Hello imap,

   Ceci est le corps de mon message

--
Best regards,
 Thomas.
```

Le code source du script chargé de récupérer les informations est le suivant :

Listing 12.16 : imap_5.php

```
<html>
<head><title>Exemple IMAP</title></head>
<body>
<?php
$mbox   = imap_open("{mail.toutestfacile.com:110/pop3}",
                    "imap@toutestfacile.com", "pipo");
$header = imap_headerinfo($mbox, 8);
echo "message_id:".$header->message_id."<br />\n";
echo "Date:".$header->Date."<br />\n";
```

```
echo "date:".$header->date."<br />\n";
echo "Subject:".$header->Subject."<br />\n";
echo "subject:".$header->subject."<br />\n";
echo "Recent:".$header->Recent."<br />\n";
echo "Unseen:".$header->Unseen."<br />\n";
echo "Answered:".$header->Answered."<br />\n";
echo "Deleted:".$header->Deleted."<br />\n";
echo "Flagged:".$header->Flagged."<br />\n";
echo "toaddress:".$header->toaddress."<br />\n";
echo "to[0]->mailbox:".$header->to[0]->mailbox."<br />\n";
echo "to[0]->host:".$header->to[0]->host."<br />\n";
echo "to[1]->mailbox:".$header->to[1]->mailbox."<br />\n";
echo "to[1]->host:".$header->to[1]->host."<br />\n";
echo "fromaddress:".$header->fromaddress."<br />\n";
echo "from[0]->personal:".$header->from[0]->personal."<br />\n";
echo "from[0]->mailbox:".$header->from[0]->mailbox."<br />\n";
echo "from[0]->host:".$header->from[0]->host."<br />\n";
echo "ccaddress:".$header->ccaddress."<br />\n";
echo "cc[0]->mailbox:".$header->cc[0]->mailbox."<br />\n";
echo "cc[0]->host:".$header->cc[0]->host."<br />\n";
echo "reply_toaddress:".$header->reply_toaddress."<br />\n";
echo "reply_to[0]->personal:".$header->reply_to[0]->personal."<br />\n";
echo "reply_to[0]->mailbox:".$header->reply_to[0]->mailbox."<br />\n";
echo "reply_to[0]->host:".$header->reply_to[0]->host."<br />\n";
echo "senderaddress:".$header->senderaddress."<br />\n";
echo "sender[0]->personal:".$header->sender[0]->personal."<br />\n";
echo "sender[0]->mailbox:".$header->sender[0]->mailbox."<br />\n";
echo "sender[0]->host:".$header->sender[0]->host."<br />\n";
echo "udate:".$header->udate."<br />\n";
imap_close($mbox);
?>
</body>
</html>
```

Le résultat obtenu est le suivant :

```
message_id:<10913313113.20020616220414@toutestfacile.com>
Date:Sun, 16 Jun 2002 22:04:14 -0400
date:Sun, 16 Jun 2002 22:04:14 -0400
Subject:Ceci est le sujet du mail
subject:Ceci est le sujet du mail
Recent:N
Unseen:
Answered:
Deleted:
Flagged:
toaddress:imap@toutestfacile.com,copie@toutestfacile.com
to[0]->mailbox:imap
to[0]->host:toutestfacile.com
to[1]->mailbox:copie
to[1]->host:toutestfacile.com
fromaddress:ToutEstFacile
```

```
from[0]->personal:ToutEstFacile
from[0]->mailbox:webmaster
from[0]->host:toutestfacile.com
ccaddress:copieCC@toutestfacile.com
cc[0]->mailbox:copieCC
cc[0]->host:toutestfacile.com
reply_toaddress:ToutEstFacile
reply_to[0]->personal:ToutEstFacile
reply_to[0]->mailbox:webmaster
reply_to[0]->host:toutestfacile.com
senderaddress:ToutEstFacile
sender[0]->personal:ToutEstFacile
sender[0]->mailbox:webmaster
sender[0]->host:toutestfacile.com
udate:1024279454
```

Outre celles vues dans l'exemple, d'autres informations peuvent être disponibles. En voici une liste complète :

- `remail` : redirection automatique de mails.
- `date`, `Date` : date du message.
- `udate` : date au format UNIX.
- `subject`, `Subject` : sujet du message.
- `in_reply_to` : adresse à laquelle le message répond.
- `message_id` : identifiant du message.
- `newsgroups`.
- `followup_to` : adresse de suivi du message.
- `references`.
- `fetchfrom` : ligne d'en-tête de '`TO:`' formatée pour tenir dans `$tailleFrom`.
- `fetchsubject` : ligne d'en-tête de '`Subject:`' formatée pour tenir dans `$tailleSubject`.
- `Recent` : '`R`' si le message est récent et lu, '`N`' s'il est récent mais non lu, ' ' sinon.
- `Unseen` : '`U`' si le message est non lu et non récent. ' ' si le message a été marqué lu ou s'il est non lu et récent.
- `Answered` : '`A`' si le message a été répondu, ' ' sinon.
- `Deleted` : '`D`' si le message est effacé, ' ' sinon.
- `Draft` : '`X`' si le message est un brouillon, ' ' sinon.
- `Flagged` : '`F`' si le message est marqué, ' ' sinon.
- `toaddress` : ligne d'en-tête de '`TO:`' limitée à 1024 caractères.
- `to[]` : tableau de toutes les adresses de '`TO:`' sous forme d'objet.
- `to[]->personal` : nom de la personne.
- `to[]->adl` : at domain source route (???).

- `to[]->mailbox` : partie précédant '`@`'.
- `to[]->host` : partie suivant '`@`'.
- `fromaddress` : ligne d'en-tête de '`FROM:`' limitée à 1024 caractères.
- `from[]` : tableau de toutes les adresses de '`FROM:`' sous forme d'objet.
- `from[]->personal` : nom de la personne.
- `from[]->adl` : at domain source route (???).
- `from[]->mailbox` : partie précédant '`@`'.
- `from[]->host` : partie suivant '`@`'.
- `ccaddress` : ligne d'en-tête de '`CC:`' limitée à 1024 caractères.
- `cc[]` : tableau de toutes les adresses de '`CC:`' sous forme d'objet.
- `cc[]->personal` : nom de la personne.
- `cc[]->adl`: at domain source route (???).
- `cc[]->mailbox` : partie précédant '`@`'.
- `cc[]->host` : partie suivant '`@`'.
- `bccaddress` : ligne d'en-tête de '`BCC:`' limitée à 1024 caractères.
- `bcc[]` : tableau de toutes les adresses de '`BCC:`' sous forme d'objet.
- `bcc[]->personal` : nom de la personne.
- `bcc[]->adl` : at domain source route (???).
- `bcc[]->mailbox` : partie précédant '`@`'.
- `bcc[]->host` : partie suivant '`@`'.
- `reply_toaddress` : ligne d'en-tête de '`Reply_to:`' limitée à 1024 caractères.
- `reply_to[]` : tableau de toutes les adresses de '`Reply_to:`' sous forme d'objet.
- `reply_to[]->personal` : nom de la personne.
- `reply_to[]->adl` : at domain source route (???).
- `reply_to[]->mailbox` : partie précédant '`@`'.
- `reply_to[]->host`: partie suivant '`@`'.
- `senderaddress` : ligne d'en-tête de '`Sender:`' limitée à 1024 caractères.
- `sender[]` : tableau de toutes les adresses de '`Sender:`' sous forme d'objet.
- `sender[]->personal` : nom de la personne.
- `sender[]->adl` : at domain source route (???).
- `sender[]->mailbox` : partie précédant '`@`'.
- `sender[]->host` : partie suivant '`@`'.
- `return_path[]` : tableau de toutes les adresses de '`Return-path:`' sous forme d'objet.
- `return_path[]->personal` : nom de la personne.
- `return_path[]->adl` : at domain source route (???).

- `return_path[]->mailbox` : partie précédant '@'.
- `return_path[]->host` : partie suivant '@'.

Lecture des messages

Certes, accéder à la liste des messages est une chose importante, mais accéder à leur contenu l'est encore plus ! Vous disposez là aussi de nombreuses fonctions ; la première d'entre elles est `imap_body()`.

imap_body()

Retourne le corps d'un message.

Syntaxe	`string imap_body(resource $identifiantBal, int $numeroMsg [, int $mode]);`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$numeroMsg`	Numéro du message à ouvrir (le premier porte l'indice 1).
`$mode`	Combinaison par `OU` logique des options : *FT_UID* si le numéro du message est son UID. *FT_PEEK* afin de ne pas indiquer le message comme étant lu (s'il n'est pas déjà marqué). *FT_INTERNAL* afin que la chaîne retournée soit dans le format 'interne'.
`retour`	Le corps du message.

Il faut toutefois garder à l'esprit qu'un message peut être composé de plusieurs parties (généralement le corps du document que l'on peut récupérer avec `imap_body()`, mais aussi des fichiers joints).

Aussi, avant d'aller plus loin, vous devez connaître la structure du message. Comme par miracle, il y a justement une fonction qui permet cela.

imap_fetchStructure()

Lit la structure d'un message. L'objet retourné contient l'enveloppe, la date, la taille, les drapeaux et la structure du corps, ainsi que ces informations pour les composantes MIME du message.

Syntaxe	`object imap_fetchStructure(resource $identifiantBal, int $numeroMsg [, int $mode])`
`$identifiantBal`	Identifiant retourné par `imap_open()`.
`$numeroMsg`	Indice du message à ouvrir (le premier porte l'indice 1).

`$mode`	`FT_UID` (si le numéro du message est son UID).
`retour`	Un objet contenant les attributs décrits ci-après :

- `type` : le type de contenu qui peut prendre l'une des valeurs suivantes :
  - `TYPETEXT` s'il s'agit d'un texte non formaté ;
  - `TYPEMULTIPART` s'il s'agit d'un message avec plusieurs parties .
  - `TYPEMESSAGE` ;
  - `TYPEAPPLICATION` ;
  - `TYPEAUDIO` s'il s'agit d'un fichier audio ;
  - `TYPEIMAGE` s'il s'agit d'un fichier image ;
  - `TYPEVIDEO` s'il s'agit d'un fichier vidéo ;
  - `TYPEOTHER` s'il s'agit d'un type inconnu.
- `encoding` : le type d'encodage qui peut prendre l'une des valeurs `ENCBASE64`, `ENCQUOTEDPRINTABLE`, `ENCOTHER` ;
- `ifsubtype` : `TRUE` si un sous-type MIME est défini (attribut `subtype`) ;
- `subtype` : sous-type MIME ;
- `ifdescription` : `TRUE` si une description est définie (attribut `description`) ;
- `description` : description du contenu ;
- `ifid` : `TRUE` si une chaîne d'identification est définie (attribut `id`) ;
- `id` : chaîne d'identification ;
- `lines` : nombre de lignes ;
- `bytes` : nombre d'octets ;
- `ifdisposition` : `TRUE` si l'attribut `disposition` est défini ;
- `disposition` : chaîne de disposition (ex. : `inline`, `attachment`) ;
- `ifdparameters` : `TRUE` si l'attribut `dparameters` est défini ;
- `dparameters` : un tableau d'objets où chacun des objets possède les attributs `attribute` et `value` correspondant aux paramètres de `Content-diposition` de l'en-tête MIME ;
- `ifparameters` : `TRUE` si l'attribut `parameters` est défini ;
- `parameters` : un tableau d'objets où chacun des objets possède les attributs `attribute` et `value` ;
- `parts` : un tableau d'objets identique au niveau de la structure à l'objet supérieur décrivant chacune des parties du message.

Voici un script montrant quelques résultats :

Listing 12.17 : imap_fetchstructure.php

```
<?php
    $bal = imap_open("{mail.monsite.com:143/imap}INBOX",
                     "monlogin", "monpassword");
```

```
    $objet = imap_fetchstructure($bal, 1);
    echo "\$objet->type:".$objet->type."<br />\n";
    echo "\$objet->encoding:".$objet->encoding."<br />\n";
    echo "\$objet->ifsubtype:".$objet->ifsubtype."<br />\n";
    echo "\$objet->subtype:".$objet->subtype."<br />\n";
    echo "\$objet->ifdescription:".
         $objet->ifdescription."<br />\n";
    echo "\$objet->ifid:".$objet->ifid."<br />\n";
    echo "\$objet->lines:".$objet->lines."<br />\n";
    echo "\$objet->bytes:".$objet->bytes."<br />\n";
    echo "\$objet->ifdisposition:".$objet->ifdisposition.
         "<br />\n";
    echo "\$objet->ifdparameters:".$objet->ifdparameters.
         "<br />\n";
    echo "\$objet->ifparameters:".$objet->ifparameters."<br />\n";
    echo "\$objet->parameters[0]->attribute:".
         $objet->parameters[0]->attribute."<br />\n";
    echo "\$objet->parameters[0]->value:".":".
         $objet->parameters[0]->value."<br />\n";
    echo "\$objet->parameters[1]->attribute:".
         $objet->parameters[1]->attribute."<br />\n";
    echo "\$objet->parameters[1]->value:".
         $objet->parameters[1]->value."<br />\n";
    imap_close($bal);
?>
```

dont un résultat pourrait être :

```
$objet->type:0
$objet->encoding:1
$objet->ifsubtype:1
$objet->subtype:PLAIN
$objet->ifdescription:0
$objet->ifid:0
$objet->lines:3
$objet->bytes:23
$objet->ifdisposition:0
$objet->ifdparameters:0
$objet->ifparameters:1
$objet->parameters[0]->attribute:charset
$objet->parameters[0]->value::us-ascii
$objet->parameters[1]->attribute:format
$objet->parameters[1]->value:flowed
```

S'il s'était agi d'un message contenant plusieurs parties, alors nous aurions eu `type=1` (car `TYPEMULTIPART=1`).

Si vous ne souhaitez extraire que la structure d'une partie du message, vous disposez de la fonction `imap_bodyStruct()`.

imap_bodyStruct()

Récupère la structure d'une partie du message.

Syntaxe	`object imap_bodyStruct(resource $identifiantBal, int $numeroMsg, int $section)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$numeroMsg`	Numéro du message à ouvrir (le premier porte l'indice 1).
`$section`	Section du courrier à ouvrir (le corps du message porte l'indice 1).
`retour`	La structure de la partie du message demandée.

Pour accéder à une partie quelconque d'un message, vous devez faire appel à `imap_fetchBody()`.

imap_fetchBody()

Retourne le contenu d'une des parties du message.

Syntaxe	`string imap_fetchbody(resource $identifiantBal, int $numeroMsg, int $numeroPartie [, int $mode])`
`$identifiantBal`	Identifiant retourné par `imap_open()`.
`$numeroMsg`	Indice du message à ouvrir (le premier porte l'indice 1).
`$numeroPartie`	Indice de la partie à retourner (le corps du message porte l'indice 1).
`$mode`	Combinaison par `OU` logique des options : *FT_UID* si le numéro du message est son `UID`. *FT_PEEK* afin de ne pas indiquer le message comme étant lu (s'il n'est pas déjà marqué). *FT_INTERNAL* afin de retourner le résultat dans le format "interne".
`retour`	La partie du message désirée.

L'exemple suivant est en deux fichiers : le premier liste les messages dans la boîtes à lettres et propose des liens vers le second de la forme `imap_4.php?no=<valeur>`, afin d'afficher le contenu du message (le corps uniquement, les parties attachées étant ici ignorées).

Listing 12.18 : imap3.php

```
<html>
<head><title>Exemple IMAP</title></head>
<body>
<?php
$mbox = imap_open("{mail.monsite:110/pop3}", "monlogin", "monpassword");
echo "<p><h1>Entetes de mail dans INBOX</h1>\n";
$headers = imap_headers($mbox);
```

```
if (!$headers) {
    echo "Erreur !\n";
} else {
    while (list ($key,$val) = each ($headers)) {
        echo "<a href=\"imap4.php?no=".($key+1)."\">".
             $val."</a><br>\n";
    }
}
imap_close($mbox);
?>
</body>
</html>
```

Le script suivant ouvre donc une connexion à un serveur POP3, puis extrait l'en-tête du mail à ouvrir afin de récupérer des informations sur l'expéditeur et, enfin, écrit le contenu du message.

Listing 12.19 : imap4.php

```
<html>
<head><title>Exemple IMAP</title></head>
<body>
<?php
$mbox   = imap_open("{mail.monsite.com:110/pop3}", "monlogin", "monpassword");
$header = imap_headerInfo($mbox, $no);
$from   = $header->from;
echo "Message de:".$from[0]->personal.
     " [".$from[0]->mailbox."@".$from[0]->host."]<br />";
$text = imap_fetchBody($mbox, $no, 1);
echo $text;
imap_close($mbox);
?>
</body>
</html>
```

dont le résultat attendu pourrait être le suivant :

```
<html>
<head><title>Exemple IMAP</title></head>
<body>
Message de: ToutEstFacile[webmaster@toutestfacile.com]<br />Ceci est un test.

Cordialement,
Thomas.
</body>
</html>
```

Recherche et tri des messages

imap_search()

Cette fonction effectue une recherche sur les messages de la boîte à lettres.

Syntaxe	`array imap_search(resource $identifiantBal, string $criteres, int $option)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$criteres`	Critères de recherche détaillés ci-après.
`$option`	`SE_UID` pour retourner les identifiants plutôt que le numéro de séquence.
`retour`	Tableau indexé des numéros de séquences ou des identifiants des messages correspondant aux critères de sélection.

Les critères peuvent être combinés en les séparant par des espaces (ce qui aura pour effet de rechercher les messages répondant au critère1 ET au critère2, etc.). Chaque critère est composé d'un des mots-clés suivants, éventuellement suivi d'une valeur (qui devra être entre guillemets) :

- `ALL` : fait une recherche dans tous les champs de tous les messages.

Critères liés aux drapeaux :

- `ANSWERED` : pour ne chercher que parmi les messages marqués "répondus".
- `UNANSWERED` : pour ne chercher que parmi les messages marqués "non répondus".
- `DELETED` : pour ne chercher que parmi les messages marqués "effacés".
- `UNDELETED` : pour ne chercher que parmi les messages non marqués "effacés".
- `FLAGGED` : pour ne chercher que parmi les messages marqués (importants).
- `UNFLAGGED` : pour ne chercher que parmi les messages non marqués.
- `NEW` : pour ne chercher que parmi les nouveaux messages.
- `OLD` : pour ne chercher que parmi les anciens messages.
- `RECENT` : pour ne chercher que parmi les messages marqués "récents".
- `SEEN` : pour ne chercher que parmi les messages marqués "lus".
- `UNSEEN` : pour ne chercher que parmi les messages marqués "non lus".

Critères liés aux adresses e-mail :

- `BCC` "adresse" : recherche dans le champ *BCC*.
- `CC` "adresse" : recherche dans le champ *BCC*.
- `FROM` "adresse" : recherche dans le champ *FROM*.
- `TO` "adresse" : recherche dans le champ *TO*.

Critères liés au texte du message :

- SUBJECT "texte" : recherche dans le champ *SUBJECT*.
- BODY "texte" : pour rechercher dans le corps des messages.
- TEXT "texte" : pour rechercher les messages contenant ce texte.
- KEYWORD "texte" : pour rechercher les messages ayant ce mot-clé.
- UNKEYWORD "texte" : pour rechercher les messages n'ayant pas ce mot-clé.

Critères liés à la date :

- BEFORE "date" : pour rechercher parmi les messages antérieurs à une certaine date.
- SINCE "date" : pour rechercher parmi les messages postérieurs à une certaine date.
- ON "date" : pour rechercher parmi les messages émis à une certaine date.
- NEW : pour ne chercher que parmi les nouveaux messages.

Ce qui pourra donner par exemple :

```
imap_search($bal, "FROM \"toutestfacile.com\" SUBJECT \"Test\"");
```

Pour retourner la liste des messages envoyés depuis une adresse contenant "toutestfacile.com" et ayant dans le mot "Test" dans le sujet.

imap_sort()

Permet de retourner une liste triée de messages.

Syntaxe	`array imap_sort(resource $identifiantBal, int $criteres, boolean $inverse, [ int $mode [, string $filtre]])`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$criteres`	Au choix, l'un des critères suivants : SORTDATE : tri selon la date du message. SORTARRIVAL : tri selon la date de réception du message. SORTFROM : tri selon l'expéditeur. SORTSUBJECT : tri selon le sujet. SORTTO : tri selon le premier destinataire. SORTCC : tri selon le champ *CC*. SORTSIZE : tri selon la taille du message.
`$inverse`	TRUE pour obtenir l'ordre inverse.
`$mode`	Combinaison par OU logique des options : SE_UID pour retourner les UID plutôt que les indices. SE_NOPREFETCH pour ne pas pré-télécharger les messages.
`$filtre`	Permet de ne récupérer que certains messages ; ce filtre fonctionne comme les critères de `imap_search()`.
`retour`	Tableau indexé des identifiants de messages triés selon l'ordre choisi.

Modification des drapeaux et suppression des messages

À chaque message sont associés des drapeaux indiquant s'il a été lu, si une réponse a été envoyée, s'il est sur le point d'être effacé, etc.

La bibliothèque IMAP vous permet de modifier ces informations.

imap_setFlag_full()

Permet de marquer un ou plusieurs messages avec un drapeau spécifique.

Syntaxe	`boolean imap_setFlag_full(resource $identifiantBal, string $listeMsg, string $drapeau [, string $mode])`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$listeMsg`	Liste des numéros (ou UID) ou intervalles de numéros (ou UID) de messages séparés par une virgule (le premier porte l'indice 1). Les intervalles sont définis par le numéro de début, le caractère : et le numéro de fin.
`$drapeau`	Drapeau à retirer "`\Seen`", "`\Answered`", "`\Flagged`", "`\Deleted`", "`\Draft`", "`\Recent`". (N'oubliez pas de doubler les anti-slashes si vous définissez la chaîne entre guillemets).
`$mode`	`FT_UID` si le numéro du message est son UID.
`retour`	`TRUE` en cas de succès.

Exemple :

```
imap_setFlag_full($bal, "1,3,5:8", "\\Seen");
```

imap_clearFlag_full()

Retire un drapeau d'un ou plusieurs messages.

Syntaxe	`boolean imap_clearFlag_full(resource $identifiantBal, string $listeMsg, string $drapeau [, string $mode])`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$listeMsg`	Liste des numéros (ou UID) ou intervalles de numéros (ou UID) de messages séparés par une virgule (le premier porte l'indice 1). Les intervalles sont définis par le numéro de début, le caractère : et le numéro de fin.
`$drapeau`	Drapeau à retirer "`\Seen`", "`\Answered`", "`\Flagged`", "`\Deleted`", "`\Draft`", "`\Recent`". (N'oubliez pas de doubler les anti-slashes si vous définissez la chaîne entre guillemets).

`$mode`	`FT_UID` si le numéro du message est son UID.
`retour`	`TRUE` en cas de succès.

Exemple :

```
imap_clearFlag_full($bal, "1,3,5:8", "\\Seen");
```

Pour les messages à marquer comme étant à effacer, vous pouvez appeler directement les fonctions suivantes :

imap_delete()

Cette fonction permet de marquer un message comme étant à effacer.

Syntaxe	`boolean imap_delete(resource $identifiantBal, int $numeroMsg [, int $drapeaux])`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$numeroMsg`	Numéro du message à effacer (le premier porte l'indice 1).
`$drapeaux`	`FT_UID` (si le numéro du message est son UID).
`retour`	`TRUE` en cas de succès.

Voici un script qui permet d'effacer le message d'indice 1 :

Listing 12.20 : imap_delete.php

```
<?php
    $bal = imap_open("{mail.monsite.com:143/imap}INBOX",
                     "monlogin", "monpassword", CL_EXPUNGE);
    echo imap_delete($bal, 1);
    imap_close($bal);
?>
```

imap_undelete()

Retire le drapeau indiquant que le message est à effacer.

Syntaxe	`boolean imap_undelete(resource $identifiantBal, int $numeroMsg)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$numeroMsg`	Numéro du message dans la boîte à lettres.
`retour`	`TRUE` en cas de succès.

La suppression du message ne sera effective qu'à la fermeture de la boîte à lettres si celle-ci a été ouverte ou fermée avec l'option *CL_EXPUNGE*. Il est toutefois possible, à tout moment, de véritablement supprimer les messages marqués comme étant à supprimer avec l'option *imap_expunge()*.

imap_expunge()

Efface tous les messages marqués comme étant à effacer.

Syntaxe	`boolean imap_expunge(resource $identifiantBal)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`retour`	`TRUE` en cas de succès.

Ajout et déplacement de messages

imap_append()

Permet d'ajouter un message dans une boîte à lettres.

Syntaxe	`boolean imap_append(resource $identifiantBal, string $bal, string $message [, string $drapeaux])`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$bal`	Boîte à lettres où écrire.
`$message`	Message à écrire.
`$drapeaux`	Drapeaux écrits dans la boîte à lettres.
`retour`	`TRUE` si l'opération a pu s'effectuer, `FALSE` dans le cas contraire.

Le script suivant écrit un message dans les brouillons :

Listing 12.21 : imap_append.php

```
<?php
    $bal = imap_open("{mail.monsite.com:143/imap}INBOX",
                     "monlogin", "monpassword");
    imap_append($bal, "{mail.monsite.com:143/imap}INBOX.Drafts",
                      "TO:imap@toutestfacile.com\r\n".
                      "Subject:Cool\r\n".
                      "\r\n".
                      "Je t'écris depuis un script PHP en IMAP ! Bisous,".
                      " Moi");
    imap_close($bal);
?>
```

imap_mail_copy()

Copie certains messages dans une boîte à lettres spécifiée.

Syntaxe	`boolean imap_mail_copy(resource $identifiantBal, string $listeMsg, string $bal[, int $mode])`
`$identifiantBal`	Identifiant retourné par `imap_open()`.
`$listeMsg`	Liste des numéros (ou UID) ou intervalles de numéros (ou UID) de messages séparés par une virgule (le premier porte l'indice 1). Les intervalles sont définis par le numéro de début, le caractère `:` et le numéro de fin.
`$bal`	Boîtes à lettres de destination.
`$mode`	Combinaison par `OU` logique des options : `CP_UID` si la liste des messages contient des UID. `CP_MOVE` pour demander la suppression des messages de la boîte d'origine.
`retour`	`TRUE` si l'opération s'est effectuée sans erreur.

imap_mail_move()

Déplace certains messages dans une boîte à lettres spécifiée. (Marque le message de la boîte d'origine comme étant à effacer).

Syntaxe	`boolean imap_mail_move(resource $identifiantBal, string $listeMsg, string $bal [, int $mode])`
`$identifiantBal`	Identifiant retourné par `imap_open()`.
`$listeMsg`	Liste des numéros (ou UID) ou intervalles de numéros (ou UID) de messages séparés par une virgule (le premier porte l'indice 1). Les intervalles sont définis par le numéro de début, le caractère `:` et le numéro de fin.
`$bal`	Boîtes à lettres de destination.
`$mode`	`CP_UID` si la liste des messages contient des UID.
`retour`	`TRUE` si l'opération s'est effectuée sans erreur.

Inscription/désinscription à un serveur de nouvelles

imap_subscribe()

Permet de s'inscrire à une boîte à lettres (pour les serveurs de "news").

Syntaxe	`boolean imap_subscribe(resource $identifiantBal, string $bal)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$bal`	Boîte à lettres où s'inscrire.
`retour`	`TRUE` si l'inscription s'est faite.

imap_unsubscribe()

Permet de se désabonner d'une liste.

Syntaxe	`boolean imap_unsubscribe(resource $identifiantBal, string $bal)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$bal`	Boîte à laquelle l'utilisateur doit être désinscrit.
`retour`	`TRUE` si l'opération s'est correctement déroulée.

Identifiants

imap_msgno()

Cette fonction retourne le numéro de séquence de l'UID fourni.

Syntaxe	`int imap_msgno(resource $identifiantBal, int $uid)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`retour`	L'indice du message.

imap_uid()

Retourne l'UID d'un message d'après son indice dans la boîte à lettres.

Syntaxe	`int imap_uid(resource $identifiantBal, int $numeroMsg)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$numeroMsg`	Indice du message dans la boîte aux lettres.
`retour`	L'UID du message.

Composition et décomposition d'adresses e-mail

Il est facile de construire une adresse e-mail contenant notamment le nom en toutes lettres du destinataire ou de l'expéditeur tout en s'assurant de sa conformité avec la norme RFC822.

imap_rfc822_write_address()

Retourne une adresse e-mail respectant la norme RFC822.

Syntaxe :	`string imap_rfc822_write_address(string $login, string $domain, string $nom)`
`$login`	Partie précédant le `@`.
`$domain`	Partie suivant le `@`.
`$nom`	Nom ou texte associé à l'adresse.
`retour`	Une adresse normée.

Le script suivant :

Listing 12.22 : imap_rfc822_write_address.php

```
<?php
    echo imap_rfc822_write_address("Emma", "tuvu.com", "Eh ! M'as tu vu ?");
?>
```

retournerait :

```
Eh ! M'as tu vu ? <Emma@tuvu.com>
```

À l'inverse, il est possible de décomposer une liste d'adresses.

imap_rfc822_parse_adrlist()

Cette fonction permet de récupérer les différentes parties d'une adresse e-mail respectant la norme RFC2822.

Syntaxe	`array imap_rfc822_parse_adrlist(string $adresse, string $hoteDefaut)`
`$adresse`	Chaîne de caractères de l'adresse.
`$hoteDefaut`	Nom de l'hôte par défaut.
`retour`	Un tableau d'objets ayant les attributs : `personal` : nom de la personne. `Mailbox` : partie précédant `@`. `Host` : partie suivant `@`. `Adl` : at domain source route.

Listing 12.23 : imap_rfc822_parse_adrlist.php

```
<?php
    $adresses= "Emma TUVU <emma@tuvu.com>, bible@php.com, Starsky";
    $tableau = imap_rfc822_parse_adrlist($adresses,"defaut.com");
```

```
    while(list($cle,$valeur)=each($tableau)){
        echo "personal: ".$valeur->personal."<br />\n";
        echo "mailbox : ".$valeur->mailbox."<br />\n";
        echo "host : ".$valeur->host."<br />\n";
        echo "adl : ".$valeur->adl."<br />\n";
    }
?>
```

Le résultat est :

```
personal: Emma TUVU
mailbox : emma
host : tuvu.com
adl :
personal:
mailbox : bible
host : php.com
adl :
personal:
mailbox : Starsky
host : defaut.com
adl :
```


Génération et envoi de mails

La bibliothèque IMAP permet également de générer et d'envoyer des e-mails.

imap_mail()

Cette fonction permet d'envoyer un courrier électronique.

Syntaxe	`boolean imap_mail(string $destinataire, string $sujet, string $message [, string $entete [, string $cc [, string $bcc [, string $rpath]]]])`
`$destinataire`	Adresse(s) du ou des destinataires.
`$sujet`	Sujet du message.
`$message`	Contenu du courrier électronique.
`$entete`	Pour ajouter un en-tête particulier.
`$cc`	Adresses des personnes en copie.
`$bcc`	Adresses des personnes en copie cachée.
`$rpath`	Adresse de retour des messages d'erreur.
`retour`	`TRUE` si l'opération s'est effectuée.

Un script très simple présentant cette fonction :

Listing 12.24 : imap_mail.php

```
<?php
imap_mail("destinataire@sonsite.com", "Test de mail", "Corps du message");
?>
```

imap_mail_compose()

Permet de créer un message MIME en construisant l'enveloppe et le corps.

Syntaxe	`string imap_mail_compose(array $enveloppe, array $corps)`
`$enveloppe`	Tableau associatif avec les clés : `from`, `to`, `cc`, `bcc`, `remail`, `return-path`, `date`, `reply-to`, `in_reply_to`, `subject`, `message_id`, `custom_headers`.
`$corps`	Tableau de tableaux, chacun des sous-tableaux pouvant avoir les clés `type`, `encoding`, `subtype`, `description`, `contents.data`, `id`, `charset`, `description.type`, `lines`, `bytes`, `md5`.
`retour`	Message au format MIME.

Coder / décoder

imap_8bit()

Convertit une chaîne 8 bits en chaîne imprimable selon la RFC2045, et coupe donc les lignes de plus de 75 caractères.

Syntaxe	`string imap_8bit(string $chaine)`
`$chaine`	Chaîne de caractères à transformer.
`retour`	Une chaîne conforme à la spécification RFC2045.

RFC2045 en anglais
Pour s'amuser et lire la langue concurrente de celle de Molière, pour pourrez trouver la RFC2045 à l'adresse : **http://www.faqs.org/rfcs/rfc2045.html**.

imap_qprint()

Convertit un caractère imprimable en chaîne 8 bits.

Syntaxe	`string imap_qprint(string $chaine)`
`$chaine`	Chaîne de caractères à convertir.
`retour`	Chaîne de caractères convertie.

imap_base64()

Décode un texte codé en base64.

Syntaxe	`string imap_base64(string $chaine)`
`$chaine`	Chaîne de caractères à décoder.
`retour`	Chaîne décodée.

imap_binary()

Convertit une chaîne 8 bits en chaîne base64.

Syntaxe	`string imap_binary(string $chaine)`
`$chaine`	Chaîne de caractères à convertir.
`retour`	Chaîne convertie.

imap_utf7_decode()

Décode une chaîne UTF-7 en chaîne 8 bits.

Syntaxe	`string imap_utf7_decode(string $chaine)`
`$chaine`	Chaîne de caractères à décoder.
`retour`	Chaîne décodée.

imap_utf7_encode()

Convertit une chaîne 8 bits en chaîne UTF-7. Cela sert pour encoder les noms de boîtes à lettres contenant des caractères spéciaux en dehors de la plage ASCII des caractères imprimables.

Syntaxe	`string imap_utf7_encode(string $chaine)`
`$chaine`	Chaîne à modifier.
`retour`	Chaîne modifiée.

imap_utf8()

Convertit une chaîne de caractères en UTF-8.

Syntaxe	`string imap_utf8(string $chaine)`
`$chaine`	Chaîne à modifier.
`retour`	Chaîne modifiée.

imap_mime_header_decode()

Décode les parties de l'en-tête MIME.

Syntaxe	`array imap_mime_header_decode(string $entete)`
`$entete`	En-tête codé.
`retour`	Un tableau avec deux clés, `charset` et `text`, qui est l'en-tête décodé.

Gérer les erreurs

imap_alert()

Cette fonction permet de connaître l'ensemble des alertes IMAP apparues depuis le dernier accès ou depuis le dernier effacement de la pile d'alertes. Une fois l'appel à cette fonction effectué, la pile est vidée.

Syntaxe	`array imap_alerts(void)`
`retour`	Tableau de tous les messages d'alerte.

imap_errors()

Cette fonction permet de connaître l'ensemble des erreurs IMAP apparues depuis le dernier accès ou depuis le dernier effacement de la pile d'erreurs. Une fois l'appel à cette fonction effectué, la pile est vidée.

Syntaxe	`array imap_errors(void)`
`retour`	Tableau de tous les messages d'erreur.

imap_last_error()

Retourne la dernière erreur survenue (s'il y en a eu une) lors du dernier appel.

Syntaxe `string imap_last_error(void)`
`retour` Message d'erreur.

Résultat surprenant
Les quelques tests effectués n'ont pas véritablement permis d'obtenir un message d'erreur.

12.3. Application d'exemple : le webmail

Progressivement, voici la construction d'une interface webmail minimaliste.

La première des pages permettra d'entrer le nom du serveur IMAP ou POP, le login de l'utilisateur et le mot de passe associé.

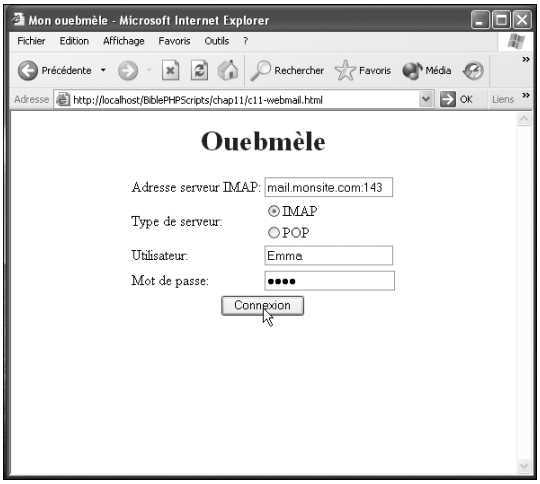


Figure 12.4 : *Interface de connexion*

Donc, le code source est le suivant :

Listing 12.25 : webmail.html

```
<html>
  <head><title>Mon ouebmèle</title></head>
  <body>
    <center>
      <p><font color="blue"><h1>Ouabmèle</h1></font></p>
      <form action="webmail1.php">
      <table>
```

```
        <tr>
          <td>Adresse serveur IMAP:</td>
          <td><input type="text" name="serveur" /></td>
        </tr>
        <tr>
          <td rowspan="2">Type de serveur:</td>
          <td><input type="radio" name="typeserveur" value="imap" />IMAP</td>
        </tr>
        <tr>
          <td><input type="radio" name="typeserveur" value="pop" />POP</td>
        </tr>
        <tr>
          <td>Utilisateur:</td>
          <td><input type="text" name="login" /></td>
        </tr>
        <tr>
          <td>Mot de passe:</td>
          <td><input type="password" name="password" /></td>
        </tr>
        <tr>
          <td colspan="2" align="center">
              <input type="submit" value="Connexion" />
          </td>
        </tr>
      </table>
      </form>
    </center>
  </body>
</html>
```


Une fois connecté, l'utilisateur est en mesure d'avoir un aperçu des messages présents dans la boîte à lettres :

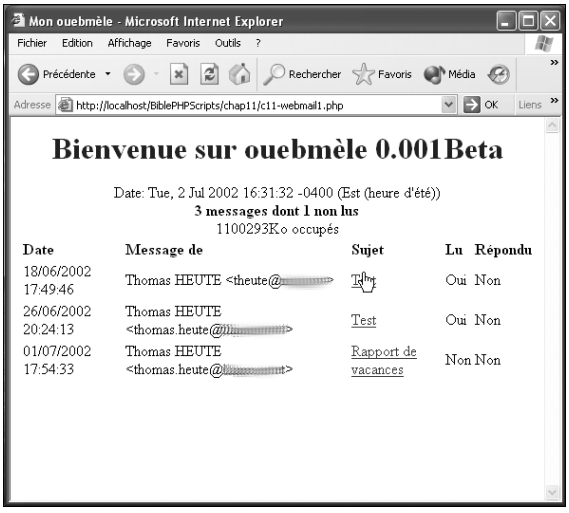


Figure 12.5 : *Liste des messages*

Listing 12.26 : webmail1.php

```
<html>
  <head><title>Mon ouebmèle</title></head>
  <body>
    <center>
      <p><font color="blue">
        <h1>Bienvenue sur ouebmèle 0.001Beta</h1>
      </font></p>
      <?php
        set_time_limit(90);
        // Connexion au serveur IMAP
        // Les données sont récupérées du formulaire par la méthode POST
        $connexion = @imap_open("{".$_POST["serveur"]."/".
                     $_POST["typeserveur"]."}INBOX",
                     $_POST["login"], $_POST["password"]) or
                     die("Impossible de se connecter :(");
        // Les informations de la boîte aux lettres INBOX
        // sont récupérées.
        $infosBal = imap_mailboxmsginfo($connexion);
        if($infosBal) {
          echo "Date: ".$infosBal->Date."<br>\n" ;
          echo "<b>".$infosBal->Nmsgs." messages dont ".
               $infosBal->Unread." non lus</b><br>\n";
          echo $infosBal->Size ."Ko occupés<br>\n" ;
        } else {
          echo "Erreur: ".imap_last_error(). "<br>\n";
        }
 ?>
        <table>
          <tr>
            <td><b>Date</b></td>
            <td><b>Message de</b></td>
            <td><b>Sujet</b></td>
            <td><b>Lu</b></td>
            <td><b>Répondu</b></td>
          </tr>
      <?php
        for ($i=1; $iNmsgs+1; $i++) {
          // Pour chacun des messages, on récupére les informations liées
          $message = imap_headerinfo($connexion, $i);
          echo "<tr>\n";
          // La date UNIX est transformée en date lisible
          echo "<td>".(date("d/m/Y H:i:s", $message->udate))."</td>\n";
          echo "<td>".(htmlspecialchars($message->fromaddress))."</td>\n";
          echo "<td><a href=\"webmail2.php?numero=$i".
                           "&serveur=".$_POST["serveur"].
                           "&typeserveur=".$_POST["typeserveur"].
                           "&login=".$_POST["login"].
                           "&password=".$_POST["password"]."\">".
                          ($message->Subject).
               "</td>\n";
```

```
          echo "<td>";
          // Les messages LU sont marqués.
          echo ($message->Unseen == 'U' || $message->Recent == 'N') ?
               "Non" : "Oui";
          echo"</td>";
          echo "<td>";
          echo ($message->Answered == 'A') ? "Oui" : "Non";
          echo"</td>";
        }
        imap_close($connexion);
 ?>
    </center>
  </body>
</html>
```

Puis l'utilisateur a la possibilité de cliquer sur un message pour voir le détail du message ainsi qu'une liste des fichiers attachés.

Listing 12.27 : webmail2.php

```
<html>
  <head><title>Mon ouebmèle</title></head>
  <body>
    <h2><font color="blue">Message</font></h2>
<?php
    // Connexion à la boîte aux lettres.
    $connexion = @imap_open("{".$_GET["serveur"]."/".
                 $_GET["typeserveur"]."}INBOX",
                 $_GET["login"], $_GET["password"])
              or die("Impossible de se connecter :(");
    $structure = imap_fetchstructure($connexion, $_GET["numero"]);
    $parties = $structure->parts;
    $fichierAttache = array();
    $i=1;
    if ($parties) {
        foreach ($parties as $partie) {
            // Pour chacune des sous-parties, on vérifie le type
            // pour l'afficher ou bien l'indiquer comme fichier attaché.
            switch ($partie->type) {
            case TYPETEXT:
                echo "<pre>".
                imap_fetchbody($connexion, $_GET["numero"], $i)."</pre>";
                break;
            default:
               $parametres = $partie->dparameters;
                foreach ($parametres as $parametre) {
                    if($parametre->attribute == "filename") {
                        $fichiersattaches[] =
                            array($i, $parametre->value);
                        break;
                    }
                }
```

```
            }
            $i++;
        }
    } else {
        // Sinon il n'y a qu'une partie
        // qui est le corps du message (texte généralement)
        echo "<pre>".imap_fetchbody($connexion,
                    $_GET["numero"], $i)."</pre>";
    }
    if ($fichiersattaches) {
?>
       <h2><font color="blue">Fichiers attachés</font></h2>
<?php
       foreach($fichiersattaches as $fichierAttache) {
            echo $fichierAttache[1]."<br />";
       }
    }
?>
  </body>
</html>
```

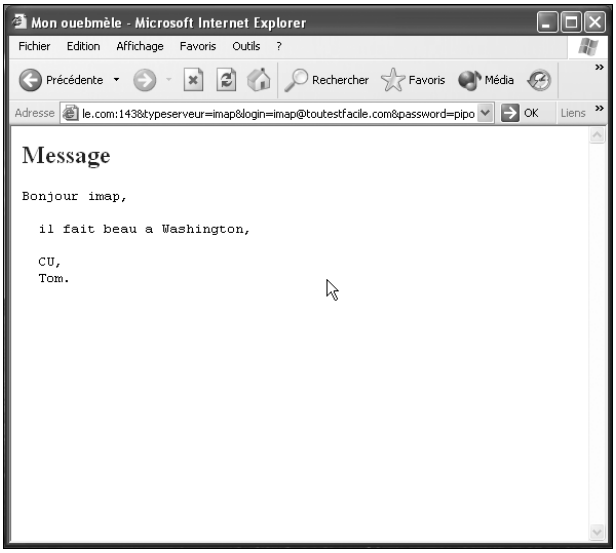


Figure 12.6 : *Exemple sans fichier attaché*

Ce webmail très simple ne montre que les bases des applications de ce genre. En quelques lignes seulement, il a été possible d'interroger un compte et de récupérer des messages. Libre à vous d'exploiter au mieux les fonctions fournies dans la bibliothèque `IMAP` pour étoffer ce script.

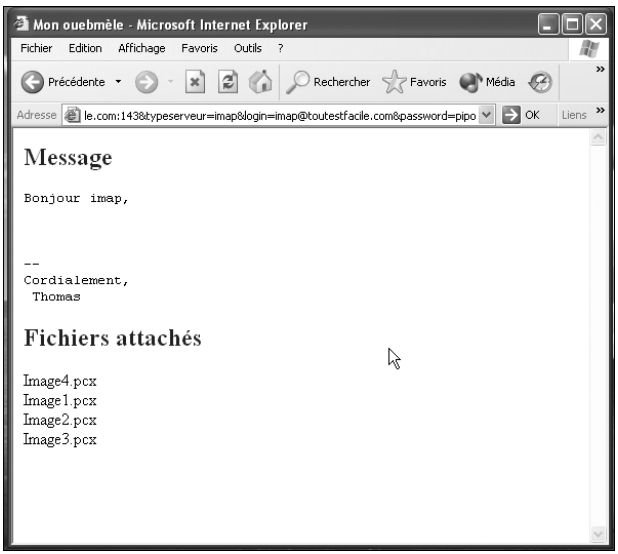


Figure 12.7 : *Exemple avec fichiers attachés*

Administration des boîtes à lettres

imap_get_quota()

Récupère les quotas d'une boîte à lettres. Il faut que l'identifiant de la boîte à lettres ait été récupéré par la fonction `imap_open()` avec les login et mot de passe de l'administrateur.

Syntaxe	`array imap_get_quota(resource $identifiantBal, string $bal)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$bal`	Nom de la boîte à lettres dont on veut les quotas (ex : `user.emma`).
`retour`	Un tableau associatif avec les clés `usage` pour connaître l'espace occupé et `limit` pour connaître la taille de la boîte à lettres.

Le script suivant est censé afficher les quotas de l'utilisateur `Emma` :

```
<?php
$bal = imap_open("{mail.monsite.com:143/imap}","loginadmin",
                  "passwordadmin",OP_HALFOPEN);
$tableau = imap_get_quota($bal, "user.emma");
if (is_array($tableau)) {
   echo "Espace occupé : " . $tableau['usage'];
   echo "Quota : " . $tableau['limit'];
}
imap_close($bal);
?>
```

imap_set_quota()

Définit les quotas d'une boîte à lettres. Il faut que l'identifiant de la boîte à lettres ait été récupéré par la fonction `imap_open()` avec les login et mot de passe de l'administrateur.

Syntaxe	`boolean imap_get_quota(resource $identifiantBal, string $bal, string $limite)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$bal`	Nom de la boîte à lettres dont on veut définir les quotas (ex. : `user.emma`).
`$limite`	Limite en Ko.
`retour`	`TRUE` si le changement de quota s'est effectué, `FALSE` sinon.

imap_createMailbox()

Crée une nouvelle boîte à lettres.

Syntaxe	`boolean imap_createMailbox(resource $identifiantBal, string $bal)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$bal`	Nom de la boîte à créer.
`retour`	`TRUE` si l'opération s'est bien effectuée, `FALSE` sinon.

Accents
Les noms de boîtes à lettres contenant des accents doivent d'abord être encodés par la fonction `imap_utf7_encode()`.

Voici un exemple créant un dossier *Test*.

Listing 12.28 : imap_createmailbox.php

```
<?php
    $bal = imap_open("{mail.monsite.com:143/imap}INBOX",
                     "monlogin", "monpassword");
    imap_createMailbox($bal, "{mail.monsite.com:143/imap}INBOX.Test");
    imap_close($bal);
?>
```

imap_deleteMailbox()

Supprime une boîte à lettres.

Syntaxe	`boolean imap_deleteMailbox(resource $identifiantBal, string $bal)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$bal`	Nom de la boîte à supprimer.
`retour`	`TRUE` si l'opération s'est bien effectuée, `FALSE` sinon.

Voici un exemple supprimant le dossier *Test*.

Listing 12.29 : imap_deletemailbox.php

```
<?php
    $bal = imap_open("{mail.monsite.com:143/imap}INBOX",
                     "monlogin", "monpassword");
    imap_deleteMailbox($bal, "{mail.monsite.com:143/imap}INBOX.Test");
    imap_close($bal);
?>
```


imap_renameMailbox()

Permet de renommer une boîte à lettres.

Syntaxe	`boolean imap_renameMailbox(resource $identifiantBal, string $ancienNom, string $nouveauNom)`
`$identifiantBal`	Identifiant tel que retourné par `imap_open()`.
`$ancienNom`	Ancien nom de la boîte à lettres.
`$nouveauNom`	Nouveau nom de la boîte à lettres.
`retour`	`TRUE` si la boîte a été renommée.

Voici un exemple d'utilisation :

Listing 12.30 : imap_renamemailbox.php

```
<?php
$bal = imap_open("{mail.monsite.com:143/imap}",
                 "monlogin","monpassword");
imap_renameMailbox($bal,"{mail.monsite.com:143/imap}.INBOX.ancien",
                   "{mail.monsite.com:143/imap}.INBOX.nouveau");
imap_close($bal);
?>
```

Chapitre 13

Les images et les animations Flash

13.1. Images (utilisation de la bibliothèque GD)

La bibliothèque GD, écrite en C, permet de créer des images aux formats JPEG, PNG, WBMP, GD et de lire un certain nombre d'autres formats (comme XPM et XPM). Auparavant, elle offrait la possibilité de créer des images au format GIF, mais cela n'est plus supporté depuis la modification de la licence d'utilisation du format GIF (licence qui ne concerne pas l'Europe). Il est cependant possible de modifier la bibliothèque GD afin de créer également des images au format GIF.

Même si vous êtes à même de dessiner de belles figures géométriques, comme des fractales, l'utilisation de cette bibliothèque vous servira certainement plutôt à tracer des diagrammes de statistiques (histogrammes, camemberts, etc.) ou à utiliser des polices d'écriture exotiques.

À la fin de ce chapitre, nous étudierons particulièrement la création des histogrammes. À travers cet exemple, nous verrons les principales fonctions disponibles grâce à cette bibliothèque.

Nous présenterons également quelques fonctions en marge de la bibliothèque GD, qui permettent de récupérer des informations sur un fichier (type, taille, etc.). Ce qui permet, par exemple, de faire une page de prévisualisation d'images.

Licence GIF (LZW)
La licence empêchant l'utilisation du format GIF n'a désormais plus court dans la plupart des pays. Pour certain pays toutefois celle-ci court encore jusqu'en juillet 2004. Il faudra donc attendre cette date pour espérer retrouver un support officiel de format GIF par la bibliothèque GD.

Installation

Sous Windows

Avec l'archive du PHP Group

Sans support GIF

Vous devrez vous assurer d'avoir le fichier *php_gd2.dll* (livré dans l'archive PHP distribuée par le PHP Group) dans votre répertoire contenant les extensions PHP, et ajouter à votre fichier *php.ini* ou décommenter une ligne

```
extension=php_gd2.dll
```

Avec les versions de PHP<4.3.2, l'archive contenait également (ou à la place) un fichier *php_gd.dll* permettant l'utilisation de la version 1 de GD.

Depuis PHP 4.3.9, le support de GIF en lecture et écriture est rétabli.

Avec support GIF

Si vous souhaitez disposer d'une librairie GD (<2.0) supportant le format GIF, vous devez télécharger un fichier *php_gd_gif.dll* (habituellement disponible sur http://www.php4win.com). Fi-

chier que vous copierez dans votre répertoire contenant les extensions PHP. Il vous suffira alors d'ajouter à votre fichier *php.ini*, une ligne

```
extension=php_gd_gif.dll
```

Avec EasyPHP

Sans support GIF

Avec EasyPHP 1.6, le support de GD (<2.0) est activé automatiquement.

Si vous souhaitez le support de GD 2 vous devrez modifier le fichier *php.ini* afin de décommenter la ligne concernant GD 2.

```
extension=php_gd2.dll
```

Avec support GIF

Si vous souhaitez le support de GD (<2.0) avec GIF, vous devez modifier le fichier *php.ini* afin de commenter la ligne concernant GD (<2.0) sans GIF, et décommenter celle concernant GD (<2.0 avec support GIF.

```
;extension=php_gd.dll
extension=php_gd_gif.dll
```

Sous Linux

L'installation de la bibliothèque `GD` nécessite (selon les besoins) la présence préalable des bibliothèques (de développement) `libpng` et `zlib` pour le format PNG, `jpeg-6b` (ou supérieur) pour le format JPEG, `freetype` et/ou `Xpm`. Ces bibliothèques sont généralement fournies avec les distributions Linux. Il suffit alors de les installer (si cela n'a pas été fait précédemment). Le plus simple consiste encore à utiliser les paquetages .rpm.


REMARQUE

Exemple avec une distribution Mandrake 9.1

Il vous suffit de "monter" le CD-ROM de la distribution (mount /mnt/cdrom), de vous déplacer dans l'arborescence (cd /mnt/cdrom/Mandrake/RPMS) et de lancer les installations :

```
rpm -U zlib1-1.1.4-5mdk.i586.rpm
rpm -U zlib1-devel-1.1.4-5mdk.i586.rpm
rpm -U libpng3-1.2.5-2mdk.i586.rpm
rpm -U libpng3-devel-1.2.5-2mdk.i586.rpm
rpm -U libjpeg62-6b-26mdk.i586.rpm
rpm -U libjpeg62-devel-6b-26mdk.i586.rpm
rpm -U freetype-1.3.1-18mdk.i586.rpm
rpm -U freetype2-2.1.3-12mdk.i586.rpm
rpm -U freetype2-devel-2.1.3-12mdk.i586.rpm
```

Vous devez, dans un premier temps, récupérer la bibliothèque GD, disponible sur le site (en anglais) **http://www.boutell.com/gd/** (elle est également disponible sur le CD-ROM fourni). Si toutefois vous souhaitez également profiter du format GIF vous devez récupérer une version modifiée sur le site (anglais) **http://www.rhyme.com.au/gd/** (elle est également disponible sur le CD-ROM fourni).Dans ce cas, vous devrez remplacer les références à `gd-2.0.15` par `gd-2.0.15gif-030801`.

Vous pouvez copier l'archive sous */usr/local/src/lib*.

Il vous suffit alors de décompresser l'archive :

```
# gunzip gd-2.0.15.tar.gz
# tar xvf gd-2.0.15.tar
```

Et de lancer la génération et l'installation :

```
# cd gd-2.0.15
# ./configure
# make
# make install
```

Vous devez à présent disposer d'un fichier *libgd.a* sous le répertoire */usr/local/lib*.

Vous devrez ensuite recompiler PHP avec l'option `--with-gd=/usr/local` pour disposer d'un GD de base (ne supportant que WBMP et éventuellement GIF). À cette option, vous pouvez ajouter :

- `--with-jpeg-dir=/usr` pour le support du format JPEG ;
- `--with-png-dir=/usr --with-zlib-dir=/usr` pour le support du format PNG.

A vous d'ajuster les chemins précisés. Ils doivent correspondre aux répertoires contenant le répertoire `lib/` contenant les fichiers `libjpeg.*`, `libpng.*`, `libz.*`, etc.

Vous pouvez vous reporter au chapitre "Prise en main" pour plus de détails sur la compilation de PHP.

Vérification

Vous pouvez vérifier le bon déroulement des opérations d'installation en appelant un simple script contenant `<?php phpinfo(); ?>`. Celui-ci devra laisser apparaître :

gd

GD Support	enabled
GD Version	2.0 or higher
GIF Read Support	enabled
GIF Create Support	enabled
JPG Support	enabled
PNG Support	enabled
WBMP Support	enabled

Figure 13.1 : *phpinfo()*

Les lignes "`Gif Read Support enabled`" et "`Gif Create Support enabled`" n'apparaîtront que si vous avez installé une version de GD supportant le format GIF.

Pour récupérer des informations sur GD vous pouvez également utiliser la fonction `gd_info()` qui vous donnera les principales caractéristiques de la version installée.

gd_info()

Retourne des informations sur la version de GD installée. Cette fonction n'existe que depuis PHP 4.3.0.

Syntaxe : `array gd_info(void)`

`retour` Un tableau avec pour index :

`GD Version` : version de GD installée
`Freetype Support` : `TRUE` si le support de Freetype est installé.
`Freetype Linkage` : Si Freetype est installé, cette valeur retourne 'with freetype', 'with TTF library' ou 'with unknown library' selon la façon dont a été installé Freetype.
`T1Lib Support` : TRUE si T1Lib est supporté.
`GIF Read Support` : TRUE si la lecture des GIF est supportée.
`GIF Create Support` : TRUE si la création de GIF est supportée.
`JPG Support` : TRUE si la manipulation de JPG est supportée.
`PNG Support` : TRUE si la manipulation de PNG est supportée.
`WBMP Support` : TRUE si la manipulation de WBMP est supportée.
`XBM Support` : TRUE si la manipulation de XBM est supportée.

Définition de l'image de base

Tout d'abord, commençons par un rapide aperçu des différents formats d'image que la bibliothèque est capable de générer.

- GIF : il s'agit d'un format de compression d'image, sans perte, limité à 256 couleurs et pouvant (depuis les dernières normes) comporter une "couleur" transparente. Il est particulièrement adapté aux dessins (images avec des contours nets et des régions de couleur uniforme).
- PNG : il s'agit d'un format de compression d'image, sans perte, pouvant supporter jusqu'à 16 millions de couleurs.
- JPEG : il s'agit d'un format de compression d'image, avec perte, pouvant supporter jusqu'à 16 millions de couleurs. Le taux de compression (et donc de perte) est ajustable, et a un impact significatif sur la taille du fichier et la qualité de l'image. Ce format est particulièrement adapté aux photos (nombreuses nuances de couleurs et peu de contours nets).
- WBMP : il s'agit du format d'image utilisé par les applications WAP, sans perte ni compression, et limité à deux couleurs.
- GD et GD 2 : sont des formats propres à la bibliothèque `GD`.

Avant d'envoyer les données liées à l'image, vous devrez prévenir le navigateur du type des données que vous lui communiquez. Il suffit pour cela d'écrire, selon les cas :

```
header("Content-type: image/gif");
header("Content-type: image/png");
header("Content-type: image/jpeg");
header("Content-type: image/vnd.wap.wbmp");
```

À vous de choisir l'en-tête correspondant en fonction du fichier que vous souhaitez fournir, et des possibilités offertes par la bibliothèque `GD` installée.


REMARQUE

Déboguer

Une fois l'en-tête envoyé au navigateur, vous ne serez plus en mesure de voir les éventuels messages d'erreur rapportés par les fonctions appelées par la suite. Pour déboguer ce genre de script, vous serez donc amené à commenter la ligne de l'en-tête ou à la déplacer pour l'appeler au tout dernier moment (juste avant d'envoyer les données).


RENVOI

Vous pouvez vous reporter à l'annexe "Les en-têtes HTTP" pour plus d'informations.

Pour construire l'image (les données à envoyer au navigateur), vous devez d'abord définir un canevas (un espace de travail avec un identifiant). Pour cela, vous pouvez soit créer une nouvelle image, soit vous resservir d'une image existante.

- Pour créer une nouvelle image, il suffit de faire appel à la fonction `imagecreate()` qui retourne un identifiant pour cette image.

imageCreate()

Crée un nouvel identifiant d'image. Pour une image de 256 couleurs.

Syntaxe	`resource imageCreate(int $largeur, int $hauteur)`
`$largeur`	Largeur de l'image.
`$hauteur`	Hauteur de l'image.
`retour`	Identifiant de l'image.

Pour une image de 200 pixels de largeur et 300 de hauteur :

```
$image = imageCreate(200,300);
```

`imageCreate()` est limité à une palette de 256 couleurs. Pour en utiliser plus, il faut créer l'image à l'aide de :

imageCreateTrueColor()

Crée un nouvel identifiant d'image. Pour une image de 16 millions de couleurs.

Syntaxe	`resource imageCreateTrueColor(int $largeur, int $hauteur)`
`$largeur`	Largeur de l'image.
`$hauteur`	Hauteur de l'image.
`retour`	Identifiant de l'image.

Pour créer une image à partir d'une image existante, en fonction du type du fichier d'image que l'on souhaite récupérer, il existe toute une panoplie de fonctions qui prennent toutes en paramètre le chemin du fichier à ouvrir :

imageCreateFromGD()

Crée un nouvel identifiant d'image à partir d'une image GD existante.

Syntaxe	`resource imageCreateFromGD(string $nomFichier)`
`$nomFichier`	Nom du fichier contenant les données devant servir de base à la nouvelle image.
`retour`	Identifiant d'image.

imageCreateFromGD2()

Crée un nouvel identifiant d'image à partir d'une image GD 2 existante.

Syntaxe	`resource imageCreateFromGD2(string $nomFichier)`
`$nomFichier`	Nom du fichier contenant les données devant servir de base à la nouvelle image.
`retour`	Identifiant d'image.

imageCreateFromGIF()

Crée un nouvel identifiant d'image à partir d'une image GIF existante.

Syntaxe	`resource imageCreateFromGIF(string $nomFichier)`
`$nomFichier`	Nom du fichier contenant les données devant servir de base à la nouvelle image.
`retour`	Identifiant d'image.

imageCreateFromJPEG()

Crée un nouvel identifiant d'image à partir d'une image JPEG existante.

Syntaxe	`resource imageCreateFromJPEG(string $nomFichier)`
`$nomFichier`	Nom du fichier contenant les données devant servir de base à la nouvelle image.
`retour`	Identifiant d'image.

imageCreateFromPNG()

Crée un nouvel identifiant d'image à partir d'une image PNG existante.

Syntaxe	`resource imageCreateFromPNG(string $nomFichier)`
`$nomFichier`	Nom du fichier contenant les données devant servir de base à la nouvelle image.
`retour`	Identifiant d'image.

imageCreateFromWBMP()

Crée un nouvel identifiant d'image à partir d'une image WBMP existante.

Syntaxe	`resource imageCreateFromWBMP(string $nomFichier)`
`$nomFichier`	Nom du fichier contenant les données devant servir de base à la nouvelle image.
`retour`	Identifiant d'image.

imageCreateFromXBM()

Crée un nouvel identifiant d'image à partir d'une image XBM existante.

Syntaxe	`resource imageCreateFromXBM(string $nomFichier)`
`$nomFichier`	Nom du fichier contenant les données devant servir de base à la nouvelle image.
`retour`	Identifiant d'image.

imageCreateFromXPM()

Crée un nouvel identifiant d'image à partir d'une image XPM existante.

Syntaxe	`resource imageCreateFromXPM(string $nomFichier)`
`$nomFichier`	Nom du fichier contenant les données devant servir de base à la nouvelle image.
`retour`	Identifiant d'image.

Une autre fonction permet de n'utiliser qu'une partie d'une image GD 2 ; sa signature est la suivante :

imageCreateFromGD2Part()

Crée un nouvel identifiant d'image à partir d'un sous-ensemble d'une image GD 2 existante.

Syntaxe	`resource imageCreateFromGD2Part(string $nomFichier, int $sourceX, int $sourceY, int $largeur, int $hauteur)`
`$nomFichier`	Nom du fichier GD 2 contenant les données devant servir de base à la nouvelle image.
`$sourceX`	Ordonnée du point supérieur gauche de la partie extraite (0 = gauche de l'image).
`$sourceY`	Abscisse du point supérieur gauche de la partie extraite (0 = haut de l'image).
`$largeur`	Largeur de la partie extraite.
`$hauteur`	Hauteur de la partie extraite.
`int`	Identifiant de l'image.

Une fois l'image créée, il suffit de l'envoyer vers le navigateur à l'aide d'une des fonctions suivantes. À la place, il est également possible de sauvegarder l'image dans un fichier en renseignant le paramètre `$nomFichier` :

imageGD()

Envoie au navigateur, ou sauvegarde dans un fichier, les données de l'image au format GD.

Syntaxe	`boolean imageGD(resource $image [, string $nomFichier])`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$nomFichier`	Nom du fichier dans lequel sauvegarder l'image.
`retour`	`FALSE` en cas d'erreur.

imageGD2()

Envoie au navigateur, ou sauvegarde dans un fichier, les données de l'image au format GD 2.

Syntaxe	`boolean imageGD2(resource $image [, string $nomFichier])`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$nomFichier`	Nom du fichier dans lequel sauvegarder l'image.
`retour`	`FALSE` en cas d'erreur.

imageGIF()

Envoie au navigateur, ou sauvegarde dans un fichier, les données de l'image au format GIF.

Syntaxe	`boolean imageGIF(resource $image [, string $nomFichier])`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$nomFichier`	Nom du fichier dans lequel sauvegarder l'image.
`retour`	`FALSE` en cas d'erreur.

imageJPEG()

Envoie au navigateur, ou sauvegarde dans un fichier, les données de l'image au format JPEG.

Syntaxe	`boolean imageJPEG(resource $image [, string $nomFichier [, int $qualite]])`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$nomFichier`	Nom du fichier dans lequel sauvegarder l'image. (Mettre une chaîne vide si vous souhaitez préciser une qualité, mais ne pas sauvegarder l'image dans un fichier).
`$qualite`	Valeur entre `1` (moins bonne qualité, plus fort taux de compression) et `100` (meilleure qualité, plus faible taux de compression). Par défaut, le facteur de qualité est d'environ `75`.
`retour`	`FALSE` en cas d'erreur.

imageWBMP()

Envoie au navigateur, ou sauvegarde dans un fichier, les données de l'image au format WBMP.

Syntaxe	`boolean imageWBMP(resource $image [, string $nomFichier [, int $couleurPrincipale]])`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$nomFichier`	Nom du fichier dans lequel sauvegarder l'image. (Mettre une chaîne vide si vous souhaitez préciser la couleur principale, mais ne pas sauvegarder l'image dans un fichier).
`$couleurPrincipale`	Couleur de premier plan (par défaut : noir) .
`retour`	`FALSE` en cas d'erreur.

image2WBMP()

Envoie au navigateur, ou sauvegarde dans un fichier, les données de l'image au format WBMP.

Syntaxe	`boolean image2WBMP(resource $image [, string $nomFichier [, int $seuil]])`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$nomFichier`	Nom du ficher dans lequel sauvegarder l'image. (Mettre une chaîne vide si vous souhaitez préciser un seuil, mais ne pas sauvegarder l'image dans un fichier).
`retour`	`FALSE` en cas d'erreur.

Pour libérer la mémoire occupée, il suffit d'appliquer la fonction `imageDestroy()` à l'image.

imageDestroy()

Libère les ressources allouées par l'image.

Syntaxe	`boolean imageDestroy(resource $image)`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`retour`	`TRUE`.

Afin de clarifier les choses, voici un exemple tout simple d'utilisation. Le script suivant affiche simplement une image au format JPEG à partir d'une image au format GIF. Ce qui,

notamment si l'image est une photo, permet, a priori, de réduire la taille de l'image transférée. En pratique, on aura tout intérêt à réaliser manuellement l'opération une bonne fois pour toutes, plutôt que d'appeler ce script chaque fois que l'image est consultée.

Listing 13.1 : gd_00.php

```
<?php

    // Précise au navigateur le type de données
    // qu'il est sur le point de recevoir
    header("Content-type: image/jpeg");

    // Prépare une nouvelle image à partir
    // d'une image au format GIF
    $image = imageCreateFromGIF("image.gif");

    // Envoie l'image au format JPEG
    // Avec un facteur de qualité de 90%
    imageJPEG($image, "", 90);

    // Libère les ressources
    // Même si, c'est surtout utile pour
    // les images intermédiaires ou bien lorsque l'image
    // est stockée dans un fichier et que le script continue
    imageDestroy($image);
?>
```

À titre d'information, si vous sauvegardez l'image obtenue, vous constaterez, qu'avec l'image d'exemple, nous avons réduit la taille des données d'un facteur 3.

L'image pourra être consultée directement en appelant l'URL de *gd_00.php* depuis le navigateur ou par le biais d'une page HTML contenant le code utilisé pour n'importe quelle image :

```
<img src="gd_00.php" />
```

La palette de couleurs

Les couleurs possèdent trois composantes : une rouge, une verte et une bleue. En informatique, chacune de ces composantes est généralement codée sur un octet, et peut donc prendre une valeur entre 0 et 255. Il est ainsi possible de définir 16 millions de couleurs. Cependant, tous les formats d'image ne sont pas capables de supporter 16 millions de couleurs : beaucoup n'utilisent qu'un seul octet pour définir la couleur d'un pixel. Il faut donc trouver le moyen de passer d'une information sur 3 octets (les trois composantes de la couleur) à une sur un seul octet. Pour cela, il suffit d'avoir une table de correspondances aussi appelée palette de couleurs.

Ainsi, pour pouvoir utiliser une couleur, il faut au préalable l'ajouter à la palette et récupérer un identifiant de couleur. Pour cela, vous devez utiliser la fonction `imageColorAllocate()`.

La signature de cette fonction est la suivante :

imageColorAllocate()

Ajoute une couleur à la palette de couleurs.

Syntaxe	`int imageColorAllocate(resource $image, int $rouge, int $vert, int $bleu)`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$rouge`	Composante rouge de l'image (comprise entre 0 et 255).
`$vert`	Composante verte de l'image (comprise entre 0 et 255).
`$bleu`	Composante bleue de l'image (comprise entre 0 et 255).
`retour`	Identifiant de couleur ou `-1` en cas d'erreur.

Notez que la première couleur ainsi définie servira de couleur de fond.

Voici quelques exemples de définition d'une couleur :

```
$blanc  = imageColorAllocate($image, 255, 255, 255);
$noir   = imageColorAllocate($image, 0, 0, 0);
$gris_1 = imageColorAllocate($image, 50, 50, 50);
$gris_2 = imageColorAllocate($image, 100, 100, 100);
$gris_3 = imageColorAllocate($image, 150, 150, 150);
$rouge  = imageColorAllocate($image, 255, 0, 0);
$vert   = imageColorAllocate($image, 0, 255, 0);
$bleu   = imageColorAllocate($image, 0, 0, 255);
$mauve  = imageColorAllocate($image, 100, 0, 100);
$jaune  = imageColorAllocate($image, 0, 100, 100);
```

Il est également possible de définir une "couleur" transparente, mais il faut toutefois, au préalable, créer un nouvel élément dans la palette avec `imageColorAllocate()`. Peu importent alors les composantes choisies, les points utilisant cette couleur seront transparents (en fait, la couleur pourrait apparaître pour les logiciels ne gérant pas la transparence).

Pour cela, il suffit d'indiquer l'identifiant de la couleur à la fonction `imageColorTransparent()`.

imageColorTransparent()

Définit un élément de la palette comme transparent.

Syntaxe	`int imageColorTransparent(resource $image [, int $couleur])`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.

`$couleur`	Identifiant de la couleur tel que retourné par la fonction `imageColorAllocate()`. Si ce paramètre est omis, c'est la couleur par défaut qui est utilisée (la première couleur allouée).
`retour`	Identifiant de la couleur rendue transparente.

Voici un morceau de script pour générer une image au format PNG, de 250 pixels sur 200, avec un fond noir :

Listing 13.2 : gd_01.php

```
<?php
  header("Content-type: image/png");
  $largeur = 250;
  $hauteur = 200;
  $image = imageCreate($largeur, $hauteur);
  $noir  = imageColorAllocate($image, 0, 0, 0);
  // c'est ici que l'on dessinera sur l'image
  imagePNG($image);
  imageDestroy($image);
?>
```

Le même script avec une couleur de fond transparente :

Listing 13.3 : gd_02.php

```
<?php
  header("Content-type: image/png");
  $largeur = 250;
  $hauteur = 200;
  $image = imageCreate($largeur, $hauteur);
  $noir  = imageColorAllocate($image, 0, 0, 0);
  imageColorTransparent($image, $noir);
  // c'est ici que l'on dessinera sur l'image
  imagePNG($image);
  imageDestroy($image);
?>
```

Recherche de couleurs dans la palette

Il est également possible de récupérer l'identifiant d'une couleur si celle-ci est disponible dans la palette (cela permet aussi de tester l'existence d'une couleur dans la palette). Pour cela, vous devrez faire appel à la fonction `imageColorExact()`.

imageColorExact()

Retourne l'identifiant de la couleur si la couleur existe dans la palette.

Syntaxe	`int imageColorExact(resource $image, int $rouge, int $vert, int $bleu)`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$rouge`	Composante rouge de la couleur désirée.
`$verte`	Composante verte de la couleur désirée.
`$bleu`	Composante bleue de la couleur désirée.
`retour`	Identifiant de la couleur précisée si elle existe dans la palette, `-1` sinon.

Il existe une variante tenant compte de la composante Alpha (transparence).

imageColorExactAlpha()

Retourne l'identifiant de la couleur si la couleur existe dans la palette, en tenant compte de la composante Alpha (transparence).

Syntaxe	`int imageColorExactAlpha(resource $image, int $rouge, int $vert, int $bleu, int $alpha)`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$rouge`	Composante rouge de la couleur désirée.
`$verte`	Composante verte de la couleur désirée.
`$bleu`	Composante bleue de la couleur désirée.
`$alpha`	Composante Alpha de la couleur désirée.
`retour`	Identifiant de la couleur précisée si elle existe dans la palette, `-1` sinon.

N'avoir à disposition que 256 couleurs lorsque l'on voudrait en utiliser 16 millions peut être contraignant. Il est toutefois possible de déterminer quelle est, dans la palette de 256 couleurs, celle qui est la plus proche de celle que nous souhaiterions utiliser.

Pour récupérer la couleur de la palette la plus proche d'une couleur que l'on définit, il existe plusieurs fonctions, qui se différencient par leur manière d'aborder la notion de "plus proche".

imageColorClosest()

Retourne l'identifiant de la couleur de la palette la plus proche de la couleur voulue. En utilisant la distance euclidienne dans un espace à trois dimensions (rouge, vert, bleu). Autrement dit `$distance = sqrt(pow($rouge2-$rouge1, 2)+pow($vert2-$vert1, 2)+pow($bleu2-$bleu1, 2))`.

Syntaxe	`int imageColorClosest(resource $image, int $rouge, int $vert, int $bleu)`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$rouge`	Composante rouge de la couleur désirée.
`$vert`	Composante verte de la couleur désirée.
`$bleu`	Composante bleue de la couleur désirée.
`retour`	Identifiant de la couleur la plus proche de la couleur désirée.

Il est également possible de tenir compte de la composante Alpha (transparence).

imageColorClosestAlpha()

Retourne l'identifiant de la couleur de la palette la plus proche de la couleur voulue.

Syntaxe	`int imageColorClosestAlpha(resource $image, int $rouge, int $vert, int $bleu, int $alpha)`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$rouge`	Composante rouge de la couleur désirée.
`$vert`	Composante verte de la couleur désirée.
`$bleu`	Composante bleue de la couleur désirée.
`$alpha`	Composante Alpha de la couleur désirée.
`retour`	Identifiant de la couleur la plus proche de la couleur désirée.

En pratique, la distance euclidienne appliquée aux composantes rouge, verte et bleue n'est pas idéale et n'offre pas toujours, de visu, le résultat attendu. Il est donc parfois préférable d'utiliser un autre algorithme.

imageColorClosestHWB()

Retourne l'identifiant de la couleur la plus proche en appuyant le calcul sur les composantes Couleur, Saturation et Brillance.

Syntaxe	`int imageColorClosestHWB(resource $image, int $rouge, int $vert, int $bleu)`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$rouge`	Composante rouge de la couleur désirée.
`$vert`	Composante verte de la couleur désirée.

`$bleu`	Composante bleue de la couleur désirée.
`$alpha`	Composante Alpha de la couleur désirée.
`retour`	Identifiant de la couleur la plus proche de la couleur désirée.

Il est également possible de demander à ce que la couleur soit créée, si jamais elle n'est pas disponible dans la palette.

imageColorResolve()

Retourne l'identifiant de la couleur si elle est dans la palette. Si la couleur n'est pas dans la palette, une nouvelle couleur est ajoutée à la palette. S'il n'y a plus de place dans la palette, c'est l'identifiant de la couleur la plus proche qui est retourné.

Syntaxe	`int imageColorResolve(resource $image, int $rouge, int $vert, int $bleu)`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$rouge`	Composante rouge de la couleur désirée.
`$vert`	Composante verte de la couleur désirée.
`$bleu`	Composante bleue de la couleur désirée.
`retour`	Identifiant de la couleur désirée ou de la couleur la plus proche de la couleur désirée.

imageColorResolveAlpha()

Retourne l'identifiant de la couleur si elle est dans la palette, en tenant compte de la composante Alpha (transparence). Si la couleur n'est pas dans la palette, une nouvelle couleur est ajoutée à la palette. S'il n'y a plus de place dans la palette, c'est l'identifiant de la couleur la plus proche qui est retourné.

Syntaxe	`int imageColorResolveAlpha(resource $image, int $rouge, int $vert, int $bleu)`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$rouge`	Composante rouge de la couleur désirée.
`$vert`	Composante verte de la couleur désirée.
`$bleu`	Composante bleue de la couleur désirée.
`$alpha`	Composante Alpha de la couleur désirée.
`retour`	Identifiant de la couleur désirée ou de la couleur la plus proche de la couleur désirée.

Pour récupérer les composantes (rouge, verte, bleue) d'un élément de la palette, il suffit d'utiliser la fonction `imageColorsForIndex()`.

imageColorsForIndex()

Retourne les composantes rouge, verte et bleue d'une couleur de la palette.

Syntaxe	`array imageColorsForIndex(resource $image, int $couleur)`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$couleur`	Identifiant de la couleur tel que retourné par les fonctions `imageColorAllocate()` et `imageColorResolve()`.
`retour`	Tableau associatif contenant les clés "`red`" contenant la composante rouge, "`green`" contenant la composante verte et "`blue`" contenant la composante bleue.

Modifier la palette

La fonction `imageColorSet()` permet de remplacer une couleur de la palette.

imageColorSet()

Remplace une couleur de la palette par une autre.

Syntaxe	`boolean imageColorSet(resource $image, int $couleur, int $rouge, int $vert, int $bleu)`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$couleur`	Identifiant de la couleur tel que retourné par les fonctions `imageColorAllocate()` et `imageColorResolve()`.
`$rouge`	Nouvelle composante rouge de la couleur désirée.
`$vert`	Nouvelle composante verte de la couleur désirée.
`$bleu`	Nouvelle composante bleue de la couleur désirée.
`retour`	`FALSE` en cas d'échec.

Il est également possible d'effectuer facilement une correction gamma sur une image avec la fonction `imageGammaCorrect()`.

imageGammaCorrect()

Effectue une correction gamma sur une image.

Syntaxe	`boolean imageGammaCorrect(resource $image, double $gammaEntree, double $gammaSortie)`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$gammaEntree`	Gamma initial de l'image (probablement 1.0).
`$gammaSortie`	Gamma corrigé.
`retour`	`TRUE`.

Une fonction permet de transformer une image *TrueColor* en image limitée à 256 couleurs :

ImageTrueColorToPalette()

Transforme une image *TrueColor* en image à palette.

Syntaxe	`void imageTrueColorToPalette(resource $image, boolean $dither, int $nbCouleurs)`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$dither`	En mettant à `TRUE` ce paramètre, cela autorise le "dithering" qui permet d'obtenir des images, certes plus granuleuses, mais les couleurs seront plus proches de celles d'origine.
`$nbCouleurs`	Nombre de couleurs que l'on souhaite dans la palette.

Supprimer des couleurs

Nous avons déjà vu les fonctions `imageColorAllocate()` et `imageColorResolve()` qui permettent d'ajouter une couleur à la palette de couleurs. Sachez qu'il est également possible de retirer des couleurs de la palette à l'aide de la fonction `imageColorDeallocate()` dont voici la signature :

imageColorDeallocate()

Supprime une couleur de la palette de couleurs.

Syntaxe	`boolean imageColorDeallocate(resource $image, int $couleur)`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.

`$couleur`	Identifiant de la couleur à supprimer, tel que retourné par les fonctions `imageColorAllocate()` et `imageColorResolve()`.
`retour`	`FALSE` en cas d'échec, `TRUE` sinon.

Nombre de couleurs dans la palette

On peut également compter le nombre de couleurs définies dans une image grâce à la fonction `imageColorsTotal()`.

imageColorsTotal()

Retourne le nombre de couleurs dans la palette de l'image.

Syntaxe	`int imageColorsTotal(resource $image)`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`retour`	Nombre de couleurs dans la palette.

Du texte dans les images

Plusieurs fonctions sont disponibles pour afficher du texte dans une image. Elles se distinguent notamment par le format de la police de caractères utilisée.

On trouve ainsi, les polices :

- GD, une police "bitmap" propre à la bibliothèque `GD` ;
- TrueType, FreeType 2, PS des polices vectorielles.

Contrairement aux polices "bitmap", les polices vectorielles gardent leur aspect quelle que soit la taille que vous leur imposez.

Police de caractères GD

`imageString()` est une fonction qui permet d'écrire horizontalement une chaîne de caractères. `imageStringUp()` est son homologue, qui permet d'écrire de bas en haut. Ces deux fonctions ont des signatures identiques :

imageString()

Écrit horizontalement une chaîne de caractères sur l'image.

Syntaxe	`void imageString(resource $image, int $police, int $x, int $y, string $texte, int $couleur)`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$police`	Identifiant de la police d'écriture à utiliser. Entre 1 et 5 s'il s'agit d'une police proposée par défaut PHP ; un nombre supérieur retourné par `imageLoadFont()` sinon.
`$x`	Abscisse du point supérieur gauche du texte.
`$y`	Ordonnée du point supérieur gauche du texte.
`$texte`	Texte à afficher.
`$couleur`	Identifiant de la couleur du texte tel que retourné par la fonction `imageColorAllocate()`.

imageStringUp()

Écrit verticalement, de bas en haut, une chaîne de caractères sur l'image.

Syntaxe	`void imageStringUp(resource $image, int $police, int $x, int $y, string $texte, int $couleur)`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$police`	Identifiant de la police d'écriture à utiliser. Entre 1 et 5 s'il s'agit d'une police proposée par défaut ; un nombre supérieur retourné par `imageLoadFont()` sinon.
`$x`	Abscisse du point supérieur gauche du texte.
`$y`	Ordonnée du point supérieur gauche du texte.
`$texte`	Texte à afficher.
`$couleur`	Identifiant de la couleur du texte tel que retourné par la fonction `imageColorAllocate()`.

Les polices par défaut (entre 1 et 5) portent les noms suivants : 1 = Tiny, 2 = Small, 3 = MediumBold, 4 = Large et 5 = Giant.

En utilisant `imageString()` et `imageStringUp()`, voici ce que l'on peut faire :

Listing 13.4 : gd_03.php

```
<?php
  header("Content-type: image/png");
  $largeur = 200;
  $hauteur = 100;
  $image = imageCreate($largeur, $hauteur);
  $transparent = imageColorAllocate($image, 255, 0, 0);
  $noir = imageColorAllocate($image, 0, 0, 0);
```

```
  imageColorTransparent($image, $transparent);

  imageString($image, 1, 10, 10, "police 1", $noir);
  imageString($image, 2, 10, 25, "police 2", $noir);
  imageString($image, 3, 10, 40, "police 3", $noir);
  imageString($image, 4, 10, 55, "police 4", $noir);
  imageString($image, 5, 10, 70, "police 5", $noir);

  imageStringUp($image, 5, 150, 100, "imageStringUp", $noir);

  imagePNG($image);
  imageDestroy($image);
?>
```

Ce script écrit cinq chaînes de caractères horizontalement utilisant les cinq polices d'écriture disponibles, puis utilise la fonction similaire permettant d'écrire verticalement.

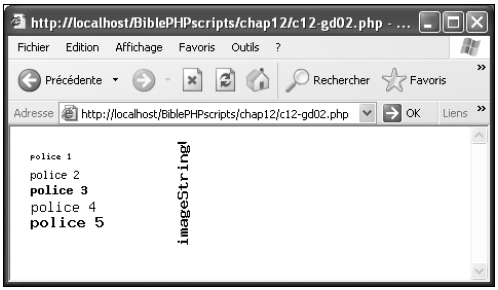


Figure 13.2 : *Utilisation d'imageString et imageStringUp*

Deux fonctions similaires, et ayant la même syntaxe que celles qui viennent d'être vues, existent et ne servent qu'à afficher le premier caractère d'une chaîne de caractères. Il s'agit de :

imageChar()

Écrit horizontalement le premier caractère d'une chaîne de caractères sur l'image.

imageCharUp()

Écrit verticalement, de bas en haut, le premier caractère d'une chaîne de caractères sur l'image.

Taille du texte

La fonction suivante retourne la hauteur en pixels d'une police d'écriture GD.

imageFontHeight()

Hauteur d'un caractère de la police d'écriture GD.

Syntaxe	`int imageFontHeight(int $police)`
`$police`	Identifiant de la police d'écriture.
`retour`	Hauteur de la police (en pixels).

Celle-ci retourne la largeur :

imageFontWidth()

Largeur d'un caractère de la police d'écriture GD.

Syntaxe	`int imageFontWidth(int $police)`
`$police`	Identifiant de la police d'écriture.
`retour`	Largeur de la police (en pixels).

Personnalisation

Il est possible d'utiliser son propre fichier de police d'écriture en utilisant la fonction suivante : `imageLoadFont()`

imageLoadFont()

Charge un fichier de police de caractères GD.

Syntaxe	`int imageLoadFont(string $nomFichier)`
`$nomFichier`	Nom du fichier de police d'écriture.
`retour`	Identifiant de police d'écriture (ayant une valeur > 5, les cinq premières étant les polices PHP par défaut).

Police de caractères TrueType

Afin de pouvoir utiliser la police TrueType, vous devez installer la bibliothèque en plus de GD.

Installation sous Linux

Récupérez la bibliothèque à l'adresse **http://www.freetype.org/**. Il s'agit d'un fichier *freetype-2.1.0.tar.gz* (également disponible sur le CD-ROM).

Après avoir copié le fichier dans un répertoire donné (ex. :*/usr/local/src/lib*), décompressez le fichier

```
# gunzip freetype-2.1.0.tar.gz
# tar xvf freetype-2.1.0.tar
# cd freetype-2.1.0
```

Puis compilez la bibliothèque :

```
# ./configure
```

Et installez-la :

```
# make install
```

Vous devez avoir maintenant une série de fichiers *libfreetype** dans le répertoire */usr/local/lib.*

Vous pouvez alors recompiler PHP avec l'option *--with-freetype-dir=/usr/local/lib*.

RENVOI

Vous pouvez vous reporter au chapitre "Prise en main" pour plus de détails.

Vérification

Vous pouvez vérifier le bon déroulement des opérations d'installation en appelant un simple script contenant `<?php phpinfo(); ?>`. Celui-ci devra laisser apparaître :

gd

GD Support	enabled
GD Version	1.6.2 or higher
FreeType Support	enabled
FreeType Linkage	with freetype
JPG Support	enabled
PNG Support	enabled
WBMP Support	enabled

Figure 13.3 : *phpinfo()*

Et qui doit donc contenir une ligne "`FreeType linkage ... with freetype`" (éventuellement "`with TTF Library`").

Utilisation

REMARQUE

Polices d'écriture
Étant donné que les images sont générées du côté du serveur, il n'est pas utile que le client ait installé les polices utilisées pour la génération des images.

Pour afficher du texte avec une telle police, il faut utiliser la fonction `imageTTFText()` dont la signature est la suivante :

imageTTFText()

Écrit une chaîne de caractères sur l'image en utilisant une police TrueType.

Syntaxe	`array imageTTFText(resource $image, int $taille, int $angle, int $x, int $y, int $couleur, string $police, string $texte)`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$taille`	Taille du texte.
`$angle`	Angle du texte en degrés (0 = texte horizontal, 90 = texte vertical de bas en haut).
`$x`	Abscisse du coin inférieur gauche du premier caractère de la chaîne.
`$y`	Ordonnée du coin inférieur gauche du premier caractère de la chaîne.
`$couleur`	Identifiant de la couleur du texte tel que retourné par la fonction `imageColorAllocate()`.
`$police`	Nom du fichier de la police TrueType
`$texte`	Texte à afficher.
`retour`	Tableau contenant quatre éléments indéterminés.

Par défaut, une opération d'"anti-aliasing" est effectuée sur le texte. Pour retirer cet effet, il suffit de faire précéder la couleur du signe '–'.

Trouver des polices d'écriture
Vous trouverez de nombreuses polices d'écriture libres de droits sur ***http://www.123fonts.com****.*

Voici un exemple de script utilisant différents angles :

Listing 13.5 : gd_04.php

```
<?php
  header("Content-type: image/png");
  $largeur = 250;
  $hauteur = 200;
  $image = imageCreate($largeur, $hauteur);
  $fond = imageColorAllocate($image, 200, 200, 200);
  $noir = imageColorAllocate($image, 0, 0, 0);

  imageTTFText($image, 20, 0, 20, 20, -$noir,
               "melanie.TTF", "Police TrueType A");
  imageTTFText($image, 20, 0, 20, 40, $noir,
               "melanie.TTF", "Police TrueType B");
  imageTTFText($image, 20, 0, 20, 60, -$noir,
               "junist.TTF", "Police TrueType C");
  imageTTFText($image, 20, 0, 20, 80, $noir,
```

```
                "junist.TTF", "Police TrueType D");
  imageTTFText($image, 20, 180, 130, 85, -$noir,
                "junist.TTF", "Police TrueType E");
  imageTTFText($image, 20, 90, 200, 150, -$noir,
                "junist.TTF", "Police TrueType F");
  imageTTFText($image, 30, 20, 5, 190, -$noir,
                "junist.TTF", "Police TrueType G");

  imagePNG($image);
  imageDestroy($image);
?>
```

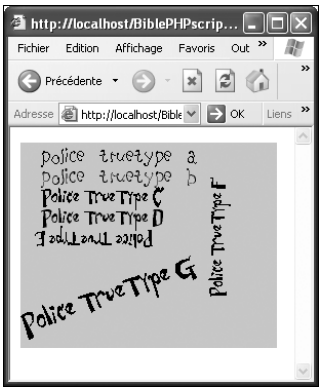


Figure 13.4 : *Utilisation d'imageTTFText*

Taille du texte

Si vous souhaitez centrer votre texte, vous devrez connaître sa taille. Pour cela, vous pouvez faire appel à :

imageTTFBBox()

Cette fonction permet de connaître les coordonnées du rectangle entourant une chaîne de caractères écrite à l'aide d'une police TrueType.

Syntaxe	`array imageTTFBBox(int $taillePolice, int $angle, string $police, string $texte)`
`$taillePolice`	Taille de la police en pixels.
`$angle`	Angle avec lequel le texte est écrit.
`$police`	Nom du fichier de la police d'écriture.
`$texte`	Texte dont on veut connaître les coordonnées du rectangle.
`retour`	Un tableau à huit éléments (abscisse du point inférieur gauche, ordonnée du point inférieur gauche, abscisse du point inférieur droit, ordonnée du point inférieur droit, abscisse du point supérieur droit, ordonnée du point supérieur droit, abscisse du point supérieur gauche, ordonnée du point supérieur gauche).

Police de caractères FreeType 2

Une autre fonction `imageFtText()` permet d'afficher du texte en utilisant des polices d'écriture de type FreeType 2. Sa syntaxe est proche de celle d'`imageTTFText()` ; un paramètre supplémentaire permet toutefois de passer d'autres informations.

imageFtText()

Écrit une chaîne de caractères sur l'image en utilisant une police FreeType 2.

Syntaxe	`array imageFtText(resource $image, int $taille, int $angle, int $x, int $y, int $couleur, string $police, string $texte, array $infos)`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$taille`	Taille du texte.
`$angle`	Angle du texte en degrés (0 = texte horizontal, 90 = texte vertical de bas en haut).
`$x`	Abscisse du coin inférieur gauche du premier caractère de la chaîne.
`$y`	Ordonnée du coin inférieur gauche du premier caractère de la chaîne.
`$couleur`	Identifiant de la couleur du texte tel que retourné par la fonction `imageColorAllocate()`.
`$police`	Nom du fichier de la police FreeType.
`$texte`	Texte à afficher.
`$infos`	Tableau associatif contenant d'autres paramètres. Pour le moment, seule la clé `"linespacing"` est prise en compte et permet de spécifier la largeur de l'interligne (coefficient par rapport à l'interligne par défaut).
`retour`	Tableau contenant quatre éléments indéterminés.

Pour écrire avec un interligne double de la normale, il suffit d'écrire :

```
imageFtText($image, $taille, $angle, $x, $y, $couleur, $police,
            $texte, array("linespacing" => 2))
```

Taille du texte

imageFTBBox()

Cette fonction permet de connaître les coordonnées du rectangle entourant une chaîne de caractères écrite à l'aide d'une police TrueType II.

Syntaxe	`array imageFTBBox(int $taillePolice, int $angle, string $police, string $texte[,array $infos])`
`$taillePolice`	Taille de la police en pixels.
`$angle`	Angle avec lequel le texte est écrit.
`$police`	Nom du fichier de la police d'écriture.
`$texte`	Texte dont nous voulons connaître les coordonnées du rectangle.
`$infos`	Informations supplémentaires telles que l'interligne.
`retour`	Un tableau à huit éléments (abscisse du point inférieur gauche, ordonnée du point inférieur gauche, abscisse du point inférieur droit, ordonnée du point inférieur droit, abscisse du point supérieur droit, ordonnée du point supérieur droit, abscisse du point supérieur gauche, ordonnée du point supérieur gauche).

Police de caractères Postscript

Installation sous Linux

Installation
Les polices Postscript ne peuvent être utilisées que si la bibliothèque `t1lib` *a été installée.*

Récupérez la bibliothèque à l'adresse **ftp://sunsite.unc.edu/pub/Linux/libs/graphics/**. Il s'agit d'un fichier *t1lib-1.3.1.tar.gz* (également disponible sur le CD-ROM).

Après avoir copié le fichier dans un répertoire donné (ex. :*/usr/local/src/lib*), décompressez-le :

```
# gunzip t1lib-1.3.1.tar.gz
# tar xvf t1lib-1.3.1.tar
# cd t1lib-1.3.1
```

Puis compilez la bibliothèque :

```
# ./configure
# make without_doc
```

Et installez-la :

```
# make
```

Vous devez avoir maintenant une série de fichiers *libt1** dans le répertoire */usr/local/lib.*

Vous pouvez alors recompiler PHP avec l'option *--with-t1lib*.

Vous pouvez vous reporter au chapitre "Prise en main" pour plus de détails sur la compilation de PHP.

Vérification

Les lignes retournées par `<?php phpinfo(); ?>` et concernant la bibliothèque `GD` doivent alors laisser apparaître l'indication "`T1Lib support enabled`".

Utilisation

Il est donc possible d'écrire du texte en utilisant une police PostScript grâce à la fonction `imagePSText()`.

imagePSText()

Écrit une chaîne de caractères sur l'image en utilisant une police Postscript.

Syntaxe	`array imagePSText(resource $image, string $texte, resource $police, int $taille, int $couleurTexte, $couleurFond, int $x, int $y [, int $largeurEspace, int $espaceCar, double $angle, int $antiAliasing])`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$texte`	Texte à afficher.
`$police`	Identifiant de police tel que retourné par la fonction `imagePSLoadFont()`.
`$taille`	Taille du texte en pixels.
`$couleurTexte`	Identifiant de la couleur du texte tel que retourné par les fonctions `imageColorAllocate()` et `imageColorResolve()`.
`$couleurFond`	Identifiant de la couleur du fond tel que retourné par les fonctions `imageColorAllocate()` et `imageColorResolve()`.
`$x`	Abscisse du coin inférieur (base) gauche du premier caractère de la chaîne (par exemple, pour la lettre p, la base se situe en dessous de la "boucle" et non au bas du "pied").
`$y`	Ordonnée du coin inférieur (base) gauche du premier caractère de la chaîne (par exemple, pour la lettre p, la base se situe en dessous de la "boucle" et non au bas du "pied").
`$largeurEspace`	Largeur (en pixels) du caractère "espace".
`$espaceCar`	Espace (en pixels) entre les caractères.
`$angle`	Angle du texte en degrés (0 = texte horizontal, 90 = texte vertical de bas en haut).
`$antiAliasing`	Nombre de paramètres à utiliser pour l'anti-aliasing (4 ou 16).
`retour`	Tableau contenant quatre éléments indéterminés.

imagePSLoadFont()

Charge une police d'écriture PostScript de type 1.

Syntaxe	`resource imagePSLoadFont(string $fileName)`
`$fileName`	Nom du fichier PostScript à charger.
`return`	Identifiant de la police d'écriture.

Taille du texte

imagePSBBox()

Cette fonction permet de connaître les coordonnées du rectangle entourant une chaîne de caractères écrite à l'aide d'une police PostScript de type 1.

Syntaxe	`array imagePSBBox(string $texte, resource $police, int $taille [,int $largeurEspace, int $espaceCar, double $angle])`
`$texte`	Texte dont vous souhaitez connaître les coordonnées du rectangle de contour.
`$police`	Identifiant de police tel que retourné par la fonction `imagePSLoadFont()`.
`$taille`	Taille du texte en pixels.
`$largeurEspace`	Largeur (en pixels) du caractère "espace".
`$espaceCar`	Espace (en pixels) entre les caractères.
`$angle`	Angle du texte en degrés (0 = texte horizontal, 90 = texte vertical de bas en haut).
`retour`	Un tableau à quatre éléments. (abscisse du point inférieur gauche, ordonnée du point inférieur gauche, abscisse du point supérieur droit, ordonnée du point supérieur droit).

Personnalisation

Il est possible de modifier les polices Postscript à l'aide des fonctions suivantes :

imagePSEncodeFont()

Permet de changer le codage vectoriel d'un caractère d'une police. Les polices d'écriture PostScript ne gèrent pas les caractères après 127. Pour utiliser une autre langue que l'anglais (contenant par exemple des caractères accentués), il peut être indispensable de modifier le codage vectoriel.

Syntaxe	`boolean imagePSEncodeFont(resource $police, string $nomFichier)`
`$police`	Identifiant de la police d'écriture à modifier tel que retourné par `imagePSLoadFont()`.
`$nomFichier`	Nom du fichier contenant le nouveau codage vectoriel.
`retour`	`TRUE`.

imagePSExtendFont()

Permet d'étendre ou de condenser une police d'écriture.

Syntaxe	`boolean imagePSExtendFont(resource $police, float $extension)`
`$police`	Identifiant de la police d'écriture à modifier tel que retourné par `imagePSLoadFont()`.
`$extension`	Indice d'extension ; s'il est inférieur à un alors la police sera condensée.
`retour`	`TRUE` si l'opération a pu se faire, `FALSE` sinon.

imagePSSlantFont()

Cette fonction permet de mettre en italique une police d'écriture.

Syntaxe	`boolean imagePSSlantFont(resource $police, float $coeff)`
`$police`	Identifiant de la police d'écriture tel que retourné par `imagePSLoadFont()`.
`$coeff`	Coefficient d'inclinaison.
`retour`	`TRUE` si l'opération a pu se faire, `FALSE` sinon.

Libération des ressources

imagePSFreeFont()

Libère la mémoire occupée par une police d'écriture PostScript de type 1.

Syntaxe	`void imagePSFreeFont(resource $police)`
`$police`	Identifiant de la police d'écriture tel que retourné par `imagePSLoadFont()`.

Dessiner des formes géométriques

Il est possible de faire ses dessins point par point, en utilisant une fonction qui permet de n'afficher qu'un seul pixel :

imageSetPixel()

Permet d'afficher un pixel d'une certaine couleur.

Syntaxe	`boolean imageSetPixel(resource $image, int $x, int $y, int $couleur)`
`$image`	Identifiant de l'image sur laquelle dessiner le pixel.
`$x`	Abscisse du pixel à dessiner.
`$y`	Ordonnée du pixel à dessiner.
`$couleur`	Identifiant de la couleur à donner au pixel.
`retour`	`TRUE`.

Grâce à cette même bibliothèque, il est aussi possible de dessiner des traits.

imageLine()

Permet de dessiner un trait.

Syntaxe	`boolean imageLine(resource $image, int $x1, int $y1, int $x2, int $y2, int $couleur)`
`$image`	Identifiant de l'image sur laquelle dessiner le trait.
`$x1`	Abscisse du point de départ.
`$y1`	Ordonnée du point de départ.
`$x2`	Abscisse du point d'arrivée.
`$y2`	Ordonnée du point d'arrivée.
`$couleur`	Identifiant de la couleur du trait, `IMG_COLOR_BRUSHED` (voir `imageSetBrush()`), `IMG_COLOR_STYLED` (voir `imageSetStyle()`) ou `IMG_COLOR_STYLEDBRUSHED`.
`retour`	`TRUE`.

Il est également possible de dessiner des traits en pointillés.

imageDashedLine()

Permet de dessiner un trait en pointillés.

Syntaxe	`boolean imageDashedLine( resource $image, int $x1, int $y1, int $x2, int $y2, int $couleur)`
`$image`	Identifiant de l'image sur laquelle dessiner le trait.
`$x1`	Abscisse du point de départ.
`$y1`	Ordonnée du point de départ.
`$x2`	Abscisse du point d'arrivée.
`$y2`	Ordonnée du point d'arrivée.
`$couleur`	Identifiant de la couleur du trait.
`retour`	`TRUE`.

Envie de vous prendre pour Picasso ? Il est possible de dessiner des arcs d'ellipses, des cercles, des polygones, des rectangles, des carrés...

imageArc()

Permet de dessiner un arc d'ellipse ou de cercle.

Syntaxe	`boolean imageArc(resource $image, int $centreX, int $centreY, int $largeur, int $hauteur, int $angleDebut, int $angleFin, int $couleur)`
`$image`	Identifiant de l'image sur laquelle dessiner l'arc d'ellipse.
`$centreX`	Abscisse du centre de l'ellipse.
`$centreY`	Ordonnée du centre de l'ellipse.
`$largeur`	Largeur de l'ellipse.
`$hauteur`	Hauteur de l'ellipse.
`$angleDebut`	Angle en degrés où commencer le tracé. 0 correspond à 3 H, 90 à 6 H, 180 à 9 H.
`$angleFin`	Angle en degrés où finir le tracé.
`$couleur`	Identifiant de la couleur de l'ellipse.
`retour`	`TRUE`.

Listing 13.6 : gd_arc.php

```
<?php
  header("Content-type: image/png");
  $largeur = 290;
```

```
  $hauteur = 50;
  $image = imageCreate($largeur, $hauteur);
  $fond = imageColorAllocate($image, 200, 200, 200);
  $noir = imageColorAllocate($image, 0, 0, 0);

  imageArc($image, 40, 25, 60, 30 , 0, 240, $noir);
  imageArc($image, 110, 25, 60, 30, 90, 180, $noir);
  imageArc($image, 180, 25, 60, 30, 180, 360, $noir);
  imageArc($image, 250, 25, 30, 30, 0, 360, $noir);

  imagePNG($image);
  imageDestroy($image);
?>
```

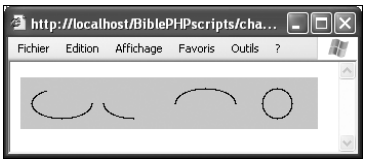


Figure 13.5 : *imageArc()*

imageFilledArc()

Permet de dessiner un arc d'ellipse ou de cercle plein.

Syntaxe	`boolean imageFilledArc(resource $image, int $centreX, int $centreY, int $largeur, int $hauteur, int $angleDebut, int $angleFin, int $couleur, int $style)`
`$image`	Identifiant de l'image sur laquelle dessiner l'arc d'ellipse.
`$centreX`	Abscisse du centre de l'ellipse.
`$centreY`	Ordonnée du centre de l'ellipse.
`$largeur`	Largeur de l'ellipse.
`$hauteur`	Hauteur de l'ellipse.
`$angleDebut`	Angle en degrés où commencer le tracé. 0 correspond à 3 H, 90 à 6 H, 180 à 9 H.
`$angleFin`	Angle en degrés où finir le tracé.
`$couleur`	Identifiant de la couleur de l'ellipse.
`$style`	Permet de définir comment remplir l'arc de cercle ; quatre valeurs combinables sont définies : `IMG_ARC_PIE`, `IMG_ARC_CHORD`, `IMG_ARC_NOFILL`, `IMG_ARC_EDGED`. Plutôt que de décrire ces constantes, nous vous suggérons de regarder l'effet de ces styles sur l'exemple qui suit.
`retour`	`TRUE`.

Listing 13.7 : gd_farc.php

```
<?php
  header("Content-type: image/png");
  $largeur = 290;
  $hauteur = 210;
  $image = imageCreate($largeur, $hauteur);
  $fond = imageColorAllocate($image, 200, 200, 200);
  $noir = imageColorAllocate($image, 0, 0, 0);

  imageFilledArc($image, 40, 25, 60, 30 , 0, 240,
                 $noir, IMG_ARC_PIE);
  imageFilledArc($image, 110, 25, 60, 30, 90, 180,
                 $noir, IMG_ARC_PIE);
  imageFilledArc($image, 180, 25, 60, 30, 180, 360,
                 $noir, IMG_ARC_PIE);
  imageFilledArc($image, 250, 25, 30, 30, 0, 360,
                 $noir, IMG_ARC_PIE);
  imageFilledArc($image, 40, 65, 60, 30 , 0, 240,
                 $noir, IMG_ARC_CHORD);
  imageFilledArc($image, 110, 65, 60, 30, 90, 180,
                 $noir, IMG_ARC_CHORD);
  imageFilledArc($image, 180, 65, 60, 30, 180, 360,
                 $noir, IMG_ARC_CHORD);
  imageFilledArc($image, 250, 65, 30, 30, 0, 360,
                 $noir, IMG_ARC_CHORD);
  // IMG_ARC_NOFILL utilise tout seul revient a
  // utiliser imageArc() au lieu de imageFilledArc()
  imageFilledArc($image, 40, 105, 60, 30 , 0, 240,
                 $noir, IMG_ARC_NOFILL);
  imageFilledArc($image, 110, 105, 60, 30, 90, 180,
                 $noir, IMG_ARC_NOFILL);
  imageFilledArc($image, 180, 105, 60, 30, 180, 360,
                 $noir, IMG_ARC_NOFILL);
  imageFilledArc($image, 250, 105, 30, 30, 0, 360,
                 $noir, IMG_ARC_NOFILL);
  imageFilledArc($image, 40, 145, 60, 30 , 0, 240,
                 $noir, IMG_ARC_EDGED);
  imageFilledArc($image, 110, 145, 60, 30, 90, 180,
                 $noir, IMG_ARC_EDGED);
  imageFilledArc($image, 180, 145, 60, 30, 180, 360,
                 $noir, IMG_ARC_EDGED);
  imageFilledArc($image, 250, 145, 30, 30, 0, 360,
                 $noir, IMG_ARC_EDGED);
  // Il est egalement possible de combiner les effets
  imageFilledArc($image, 40, 185, 60, 30 , 0, 240,
                 $noir, IMG_ARC_EDGED | IMG_ARC_NOFILL);
  imageFilledArc($image, 110, 185, 60, 30, 90, 180,
                 $noir, IMG_ARC_EDGED | IMG_ARC_NOFILL);
  imageFilledArc($image, 180, 185, 60, 30, 180, 360,
                 $noir, IMG_ARC_EDGED | IMG_ARC_NOFILL);
  imageFilledArc($image, 250, 185, 30, 30, 0, 360,
```

```
                $noir, IMG_ARC_EDGED | IMG_ARC_NOFILL);

  imagePNG($image);
  imageDestroy($image);
?>
```

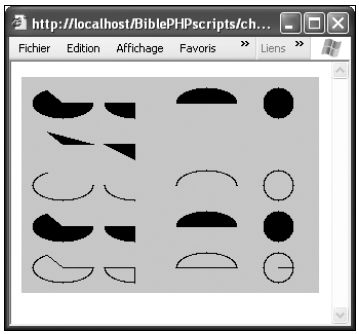


Figure 13.6 : *imageFilledArc()*

imageEllipse()

Permet de dessiner une ellipse ou un cercle (nécessite GD 2 ou plus).

Syntaxe	`boolean imageEllipse(resource $image, int $centreX, int $centreY, int $largeur, int $hauteur)`
`$image`	Identifiant de l'image sur laquelle dessiner l'ellipse.
`$centreX`	Abscisse du centre de l'ellipse.
`$centreY`	Ordonnée du centre de l'ellipse.
`$largeur`	Largeur de l'ellipse.
`$hauteur`	Hauteur de l'ellipse.
`$couleur`	Identifiant de la couleur de l'ellipse.
`retour`	`TRUE`.

imageFilledEllipse()

Permet de dessiner une ellipse ou un cercle rempli (nécessite GD 2 ou plus).

Syntaxe	`boolean imageFilledEllipse(resource $image, int $centreX, int $centreY, int $largeur, int $hauteur)`
`$image`	Identifiant de l'image sur laquelle dessiner l'ellipse.
`$centreX`	Abscisse du centre de l'ellipse.
`$centreY`	Ordonnée du centre de l'ellipse.
`$largeur`	Largeur de l'ellipse.

`$hauteur`	Hauteur de l'ellipse.
`$couleur`	Identifiant de la couleur de l'ellipse.
`retour`	`TRUE`.

Listing 13.8 : gd_fellipse.php

```
<?php
  header("Content-type: image/png");
  $largeur = 290;
  $hauteur = 50;
  $image = imageCreate($largeur, $hauteur);
  $fond = imageColorAllocate($image, 200, 200, 200);
  $noir = imageColorAllocate($image, 0, 0, 0);

  imageFilledEllipse($image, 40, 25, 60, 30, $noir);
  imageFilledEllipse($image, 110, 25, 60, 10, $noir);
  imageFilledEllipse($image, 180, 25, 10, 30, $noir);
  imageFilledEllipse($image, 250, 25, 30, 30, $noir);

  imagePNG($image);
  imageDestroy($image);
?>
```

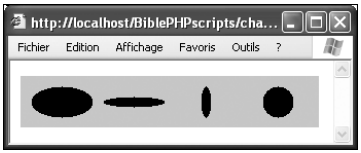


Figure 13.7 : *imageFilledArc()*

imageRectangle()

Permet de dessiner un rectangle.

Syntaxe	`boolean imageRectangle(resource $image, int $x1, int $y1, int $x2, int $y2, int $couleur)`
`$image`	Identifiant de l'image sur laquelle dessiner le rectangle.
`$x1`	Abscisse du point supérieur gauche.
`$y1`	Ordonnée du point supérieur gauche.
`$x2`	Abscisse du point inférieur droit.
`$y2`	Ordonnée du point inférieur droit.
`$couleur`	Identifiant de la couleur du rectangle.
`retour`	`TRUE`.

Listing 13.9 : gd_rectangle.php

```
<?php
  header("Content-type: image/png");
  $largeur = 151;
  $hauteur = 50;
  $image = imageCreate($largeur, $hauteur);
  $fond = imageColorAllocate($image, 200, 200, 200);
  $noir = imageColorAllocate($image, 0, 0, 0);

  imageRectangle($image, 10, 10, 70, 40, $noir);
  imageRectangle($image, 80, 10, 110, 40, $noir);
  imageRectangle($image, 120, 10, 130, 40, $noir);
  imageRectangle($image, 140, 10, 141, 40, $noir);

  imagePNG($image);
  imageDestroy($image);
?>
```

Figure 13.8 : *imageRectangle()*

imageFilledRectangle()

Permet de dessiner un rectangle plein.

Syntaxe	`boolean imageFilledRectangle(resource $image, int $x1, int $y1, int $x2, int $y2, int $couleur)`
`$image`	Identifiant de l'image sur laquelle dessiner le rectangle plein.
`$x1`	Abscisse du point supérieur gauche.
`$y1`	Ordonnée du point supérieur gauche.
`$x2`	Abscisse du point inférieur droit.
`$y2`	Ordonnée du point inférieur droit.
`$couleur`	Identifiant de la couleur du rectangle.
`retour`	`TRUE`.

Listing 13.10 : gd_frectangle.php

```
<?php
  header("Content-type: image/png");
  $largeur = 151;
  $hauteur = 50;
  $image = imageCreate($largeur, $hauteur);
```

```
  $fond = imageColorAllocate($image, 200, 200, 200);
  $noir = imageColorAllocate($image, 0, 0, 0);

  imageFilledRectangle($image, 10, 10, 70, 40, $noir);
  imageFilledRectangle($image, 80, 10, 110, 40, $noir);
  imageFilledRectangle($image, 120, 10, 130, 40, $noir);
  imageFilledRectangle($image, 140, 10, 141, 40, $noir);

  imagePNG($image);
  imageDestroy($image);
?>
```

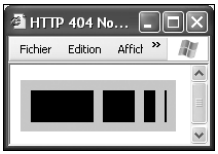


Figure 13.9 : *imageFilledRectangle()*

imagePolygon()

Permet de dessiner un polygone d'au plus 256 points.

Syntaxe :	`boolean imagePolygon(resource $image, array $points, int $nombrePoints, int $couleur)`
`$image`	Identifiant d'image sur laquelle dessiner un polygone.
`$points`	Un tableau de points de la forme `array($x1, $y1, $x2, $y2…)`
`$nombrePoints`	Nombre de points du polygone (donc généralement la moitié de la taille du tableau `$points`).
`$couleur`	Identifiant de la couleur du polygone, `IMG_COLOR_BRUSHED` (voir `imageSetBrush()`), `IMG_COLOR_STYLED` (voir `imageSetStyle()`) ou `IMG_COLOR_STYLEDBRUSHED`.
`Retour`	`TRUE`.

Listing 13.11 : gd_polygone.php

```
<?php
  header("Content-type: image/png");
  $largeur = 151;
  $hauteur = 50;
  $image = imageCreate($largeur, $hauteur);
  $fond = imageColorAllocate($image, 200, 200, 200);
  $noir = imageColorAllocate($image, 0, 0, 0);

  imagepolygon($image, array(10, 10, 70, 10, 70, 40, 10, 40), 4, $noir);
  imagePolygon($image, array(80, 10, 90, 30, 100, 15, 140, 10, 110, 40,
                             80, 35, 82, 17), 7, $noir);
```

```
  imagePNG($image);
  imageDestroy($image);
?>
```

Figure 13.10 : *imagePolygon()*

imageFilledPolygon()

Permet de dessiner un polygone plein d'au plus 256 points.

Syntaxe	`boolean imageFilledPolygon(resource $image, array $points, int $nombrePoints, int $couleur)`
`$image`	Identifiant de l'image sur laquelle dessiner un polygone.
`$points`	Un tableau de points de la forme `array($x1, $y1, $x2, $y2…)`.
`$nombrePoints`	Nombre de points du polygone (généralement la moitié de la taille du tableau `$points`).
`$couleur`	Identifiant de la couleur de remplissage du polygone ou `IMG_COLOR_TILED` (voir `imageSetTile()`).
`retour`	`TRUE`.

Listing 13.12 : gd_fpolygon.php

```
<?php
  header("Content-type: image/png");
  $largeur = 151;
  $hauteur = 50;
  $image = imageCreate($largeur, $hauteur);
  $fond = imageColorAllocate($image, 200, 200, 200);
  $noir = imageColorAllocate($image, 0, 0, 0);

  imageFilledPolygon($image, array(10, 10, 70, 10, 70, 40, 10, 40), 4, $noir);
  imageFilledPolygon($image, array(80, 10, 90, 30, 100,
                    15, 140, 10, 110, 40, 80, 35, 82, 17), 7, $noir);

  imagePNG($image);
  imageDestroy($image);
?>
```

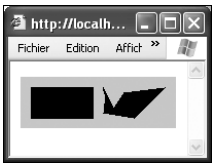


Figure 13.11 :
imageFilledPolygon()

Voici un script utilisant toutes ces fonctions :

Listing 13.13 : gd_05.php

```
<?php
  header("Content-type: image/png");
  $largeur = 290;
  $hauteur = 170;
  $image = imageCreate($largeur, $hauteur);
  $fond = imageColorAllocate($image, 200, 200, 200);
  $noir = imageColorAllocate($image, 0, 0, 0);

  imageArc($image, 40, 25, 60, 30 , 0, 240, $noir);
  imageFilledEllipse($image, 110, 25, 60, 30, $noir);
  //imageEllipse($image, 180, 40, 60, 30, $noir);

  imageFilledArc($image, 40, 65, 60, 30 , 0, 240, $noir, IMG_ARC_PIE);
  imageFilledArc($image, 110, 65, 60, 30 , 0, 240, $noir, IMG_ARC_CHORD);
  imageFilledArc($image, 180, 65, 60, 30 , 0, 240, $noir, IMG_ARC_NOFILL);
  imageFilledArc($image, 250, 65, 60, 30 , 0, 240, $noir, IMG_ARC_EDGED);

  imageRectangle($image, 10, 90, 70, 120, $noir);
  imageFilledRectangle($image, 80, 90, 140, 120, $noir);

  imagePolygon($image, array(10, 130, 20, 150, 30, 135, 70, 130,
                             40, 160, 10, 155, 12, 137), 7, $noir);
  imageFilledPolygon($image, array(80, 130, 90, 150, 100, 135, 140, 130,
                             110, 160, 80, 155, 82, 137), 7, $noir);

  imagePNG($image);
  imageDestroy($image);
?>
```

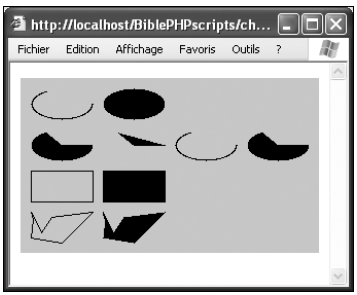


Figure 13.12 :
Utilisation des fonctions de dessin géométrique

Changer le type de tracé

Il est possible de redéfinir le type du crayon utilisé lors du tracé à l'aide de différentes méthodes.

imageSetBrush()

Permet de définir le pinceau utilisé. Le pinceau étant une image, pour l'utiliser, il faut choisir `IMG_COLOR_BRUSHED` ou `IMG_COLOR_STYLEDBRUSHED` en guise de couleur.

Syntaxe	`boolean imageSetBrush(resource $image, resource $pinceau)`
`$image`	Identifiant de l'image sur laquelle on veut redéfinir le type de tracé.
`$pinceau`	Identifiant de l'image qui servira à tracer une ligne.
`retour`	`TRUE`.

imageSetStyle()

Permet de définir le style d'un trait à l'aide d'un tableau de couleurs. Pour utiliser ce type de trait, il suffit de définir la couleur de tracé comme étant la constante `IMG_COLOR_STYLED`.

Syntaxe	`boolean imageSetStyle(resource $image, array $style)`
`$image`	Identifiant de l'image sur laquelle on veut appliquer le style de trait.
`$style`	Tableau de couleurs.
`retour`	`TRUE`.

Voici un exemple traçant une ligne composée de deux pixels bleus, puis deux rouges, puis deux verts, deux bleus, deux rouges, etc.

Listing 13.14 : gd_setstyle.php

```
<?php
header("Content-type: image/png");

$image = imagecreate(100, 100);
$blanc = imageColorAllocate($im, 255, 255, 255);
$bleu  = imageColorAllocate($im, 0, 0, 255);
$rouge = imageColorAllocate($im, 255, 0, 0);
$vert  = imageColorAllocate($im, 0, 255, 0);

$style = array($bleu, $bleu, $rouge, $rouge, $vert, $vert);
imageSetStyle($image, $style);
imageLine($image, 0, 0, 100, 100, IMG_COLOR_STYLED);
imagePNG($image);
```

```
imageDestroy($image);
?>
```

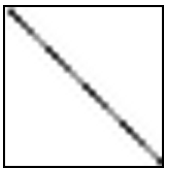


Figure 13.13 :
Le résultat agrandi

imageSetTile()

Permet de redéfinir la façon de remplir une ellipse, un rectangle… L'image fournie est répétée pour remplir la surface. Pour utiliser ce type de remplissage par la suite, il suffit de passer la constante `IMG_COLOR_TILED` comme couleur.

Syntaxe	`boolean imageSetTile(resource $image, resource $imageRemplissage)`
`$image`	Identifiant de l'image sur laquelle on veut redéfinir le remplissage.
`$imageRemplissage`	Identifiant de l'image qui servira à effectuer le remplissage de surface.
`retour`	`TRUE`.

Il est également possible de simplement définir l'épaisseur du trait :

imageSetThickness

Permet de redéfinir l'épaisseur du trait en pixels.

Syntaxe	`boolean imageSetThickness(resource $image, int $epaisseur)`
`$image`	Identifiant de l'image sur laquelle redéfinir l'épaisseur du trait.
`$epaisseur`	Épaisseur du trait.
`retour`	`TRUE`.

Listing 13.15 : gd_setthickness.php

```
<?php
header("Content-type: image/png");
$image = imageCreate(100, 100);
$fond = imageColorAllocate($image, 255, 255, 255);
$noir = imageColorAllocate($image ,0, 0, 0);
imageSetThickness($image, 10);
imageLine($image, 0, 0, 100, 100, $noir);
imagePNG($image);
?>
```

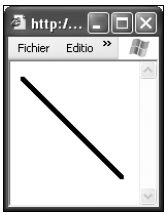


Figure 13.14 : *Le résultat*

Utiliser les couleurs

Récupérer des couleurs

Il est possible de connaître la couleur d'un pixel grâce à la fonction `imageColorAt()`.

imageColorAt()

Retourne l'identifiant de la couleur d'un pixel.

Syntaxe	`int imageColorAt(resource $image, int $x, int $y)`
`$image`	Identifiant d'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`$x`	Abscisse du pixel.
`$y`	Ordonnée du pixel.
`retour`	Identifiant de la couleur.

Copier des parties d'image

Il existe quelques fonctions qui permettent de copier des parties d'une image dans une autre.

imageCopy()

Cette fonction copie une partie d'image dans une autre.

Syntaxe	`boolean imageCopy(resource $imageDest, resource $imageSource, int $xDest, int $yDest, int $xSource, int $ySource, int $largeurSource, int $hauteurSource)`
`$imageDest`	Identifiant de l'image de destination.
`$imageSource`	Identifiant de l'image source.
`$xDest`	Abscisse sur l'image de destination.
`$yDest`	Ordonnée sur l'image de destination.

`$xSource`	Abscisse sur l'image source.
`$ySource`	Ordonnée sur l'image source.
`$largeurSource`	Largeur du morceau d'image à récupérer.
`$hauteurSource`	Hauteur du morceau d'image à récupérer.
`retour`	`TRUE`.

Listing 13.16 : gd_copy.php

```
<?php
  header("Content-type: image/png");
  $image1 = imageCreateFromJpeg("Thumbnailadrien.jpg");
  $image2 = imageCreateTrueColor(60,80);
  imageCopy($image2, $image1, 0, 0, 30, 40, 60, 80);
  imagePNG($image2);
  imageDestroy($image1);
  imageDestroy($image2);
?>
```

Figure 13.15 : *Thumbnailadrien.jpg*

Figure 13.16 : *Partie de l'image copiée*

REMARQUE

Format source et destination
Il faut impérativement copier une image TrueColor dans une autre image TrueColor, ou cela entraînera la panne du serveur. De manière générale, on ne peut copier une image que dans une autre de même type. `imageCreateFromJpeg()` *crée une image TrueColor.*

imageCopyMerge()

Cette fonction permet de copier une image dans une autre et de les fusionner selon un coefficient.

Syntaxe

```
boolean imageCopyMerge(resource $imageDest, resource
$imageSource, int $xDest, int $yDest, int $xSource, int
$ySource, int $largeurSource, int $hauteurSource, int
$coefficientTransparence)
```

`$imageDest`	Identifiant de l'image de destination.
`$imageSource`	Identifiant de l'image source.
`$xDest`	Abscisse sur l'image de destination.
`$yDest`	Ordonnée sur l'image de destination.
`$xSource`	Abscisse sur l'image source.
`$ySource`	Ordonnée sur l'image source.
`$largeurSource`	Largeur du morceau d'image à récupérer.
`$hauteurSource`	Hauteur du morceau d'image à récupérer.
`$coefficientTransparence`	C'est le coefficient qui permettra de mettre une des deux images en avant. Il est compris entre 0 et 100 ; 0 n'aura aucun effet, alors qu'utiliser 100 reviendra à utiliser la fonction `imageCopy()`.
`retour`	`TRUE`.

Le script suivant utilise la fonction `imageCopyMerge()`. Il suffit de lui passer en paramètre le coefficient de transparence pour avoir l'image *Thumbnailadriencopy.jpg* sur *avion.jpg*.

Listing 13.17 : gd_copymerge.php

```
<?php
  header("Content-type: image/png");
  $image1 = imageCreateFromJpeg("Thumbnailadriencopy.jpg");
  $image2 = imageCreateFromJpeg("avion.jpg");
  imageCopyMerge($image2, $image1, 75, 0, 0, 0, 60,
                 80, $_GET["coef"]);
  imagePNG($image2);
  imageDestroy($image1);
  imageDestroy($image2);
?>
```

Et voici une page HTML qui fait appel quatre fois à ce script en variant le coefficient de transparence.

Listing 13.18 : gd_copymerge.html

```
<html>
<head><title>ImageCopyMerge()</title></head>
<body>
  <table>
    <tr>
      <td><img src="c13-copymerge.php?coef=0" /></td>
      <td><img src="c13-copymerge.php?coef=25" /></td>
      <td><img src="c13-copymerge.php?coef=50" /></td>
      <td><img src="c13-copymerge.php?coef=100" /></td>
    </tr>
    <tr align="center">
      <td>0</td>
      <td>25</td>
```

```
      <td>50</td>
      <td>100</td>
    </tr>
  </table>
</body>
</html>
```

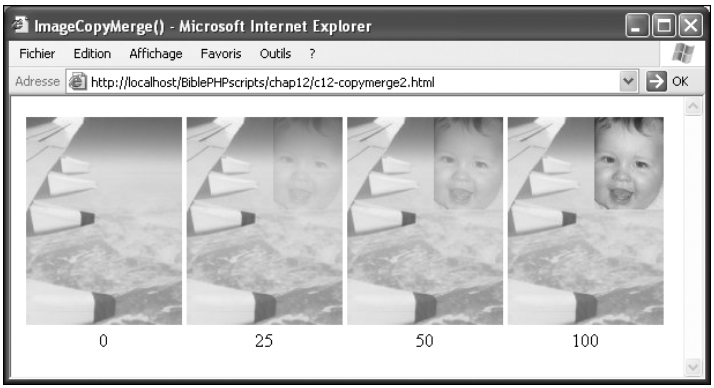


Figure 13.17 : *Quelques exemples*

imageCopyMergeGray()

Cette fonction permet de copier une image dans une autre et de les fusionner selon un coefficient, tout en transformant les pixels de destination en noir et blanc.

Syntaxe	`boolean imageCopyMergeGray(resource $imageDest, resource $imageSource, int $xDest, int $yDest, int $xSource, int $ySource, int $largeurSource, int $hauteurSource, int $coefficientTransparence)`
`$imageDest`	Identifiant de l'image de destination.
`$imageSource`	Identifiant de l'image source.
`$xDest`	Abscisse sur l'image de destination.
`$yDest`	Ordonnée sur l'image de destination.
`$xSource`	Abscisse sur l'image source.
`$ySource`	Ordonnée sur l'image source.
`$largeurSource`	Largeur du morceau d'image à récupérer.
`$hauteurSource`	Hauteur du morceau d'image à récupérer.
`$coefficient Transparence`	C'est le coefficient qui permettra de mettre une des deux images en avant. Il est compris entre 0 et 100 ; 0 n'aura aucun effet, alors qu'utiliser 100 reviendra à utiliser la fonction `imageCopy()`.
`retour`	`TRUE`.

imageCopyResized()

Cette fonction permet de recopier et de changer la taille de la sélection.

Syntaxe	`boolean imageCopyResized(resource $imageDest, resource $imageSource, int $xDest, int $yDest, int $xSource, int $ySource, int $largeurDest, int $hauteurDest, int $largeurSource, int $hauteurSource)`
`$imageDest`	Identifiant de l'image de destination.
`$imageSource`	Identifiant de l'image source.
`$xDest`	Abscisse sur l'image de destination.
`$yDest`	Ordonnée sur l'image de destination.
`$xSource`	Abscisse sur l'image source.
`$ySource`	Ordonnée sur l'image source.
`$largeurDest`	Largeur du morceau d'image après transformation.
`$hauteurDest`	Hauteur du morceau d'image après transformation.
`$largeurSource`	Largeur du morceau d'image à récupérer.
`$hauteurSource`	Hauteur du morceau d'image à récupérer.
`retour`	`TRUE`.

Le script suivant déforme une image :

Listing 13.19 : gd_copyresized.php

```
<?php
  header("Content-type: image/png");
  $image1 = imageCreateFromJpeg("avion.jpg");
  $image2 = imageCreateTrueColor(200,100);
  imageCopyResized($image2, $image1, 0, 0, 0, 0, 200, 100, 135, 180);
  imagePNG($image2);
  imageDestroy($image1);
  imageDestroy($image2);
?>
```

Figure 13.18 :
L'image de test

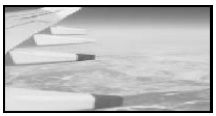


Figure 13.19 :
Le résultat

imageCopyResampled()

Cette fonction permet de recopier et de changer la taille de la sélection en utilisant une méthode non pixelisée, contrairement à `imageCopyResize()`.

Syntaxe	`boolean imageCopyResampled(resource $imageDest, resource $imageSource, int $xDest, int $yDest, int $xSource, int $ySource, int $largeurDest, int $hauteurDest, int $largeurSource, int $hauteurSource)`
`$imageDest`	Identifiant de l'image de destination.
`$imageSource`	Identifiant de l'image source.
`$xDest`	Abscisse sur l'image de destination.
`$yDest`	Ordonnée sur l'image de destination.
`$xSource`	Abscisse sur l'image source.
`$ySource`	Ordonnée sur l'image source.
`$largeurDest`	Largeur du morceau d'image après transformation.
`$hauteurDest`	Hauteur du morceau d'image après transformation.
`$largeurSource`	Largeur du morceau d'image à récupérer.
`$hauteurSource`	Hauteur du morceau d'image à récupérer.
`retour`	`TRUE`.

Le script suivant déforme une image :

Listing 13.20 : gd_copyresampled.php

```
<?php
  header("Content-type: image/png");
  $image1 = imageCreateFromJpeg("avion.jpg");
  $image2 = imageCreateTrueColor(200,100);
  imageCopyResampled($image2, $image1, 0, 0, 0, 0, 200, 100, 135, 180);
  imagePNG($image2);
  ImageDestroy($image1);
  ImageDestroy($image2);
?>
```

Figure 13.20 :
L'image de test

Figure 13.21 :
Le résultat

Pour copier une image avec un fond transparent sur une autre image, il faut copier non seulement les couleurs, mais aussi les canaux alpha. La fonction `imageAlphaBlending()` permet de le faire.

imageAlphaBlending()

Cette fonction permet de modifier la façon dont seront mélangées les couleurs d'une image lors d'une copie.

Syntaxe	`boolean imageAlphaBlending (resource $image, boolean $canauxAlpha)`
`$image`	Identifiant de l'image sur laquelle on veut modifier cette propriété.
`$canauxAlpha`	`TRUE` pour garder les canaux alpha, `FALSE` sinon. Par défaut, lors de la copie, les canaux alpha sont ignorés.
`retour`	`TRUE`.

Listing 13.21 : gd_alpha.php

```
<?php
$photo = imageCreateFromJPEG('Thumbnailadrien.jpg');
imageAlphaBlending($photo, true);
$logo = imageCreateFromPNG('logo.png');
imageCopy($photo, $logo, 10, 120, 0, 0, 50, 30);
imagePNG($photo);
imageDestroy($photo);
imageDestroy($logo);
?>
```

Figure 13.22 : *Résultat du script*

En demandant à garder les informations des canaux alpha, le fond transparent du logo a été conservé.

Il est possible depuis PHP 4.3.0 de tourner une image d'un angle quelconque, cela se fait très simplement grâce à la fonction `imagerotate()`.

ImageRotate()

Permet de tourner une image.

Syntaxe :	`resource imageRotate(resource $imageSource, float $angle, int $couleurFond)`
`$imageSource`	Image à tourner.
`$angle`	Angle selon lequel on veut tourner l'image (dans le sens inverse des aiguilles d'une montre)
`$couleurFond`	Couleur de remplissage du fond.
`retour`	L'image tournée de l'angle voulu avec un fond de couleur `$couleurFond`.

Voici un petit exemple de rotation d'une image de 30 degrés.

Listing 13.22 : gd_imagerotate.php

```
<?php
header("Content-type: image/jpeg");
$image = imageCreateFromJPEG("niki.jpg");
$fondbleu = imageColorAllocate($image, 100, 100, 255);
$image = imageRotate($image,30,$fondbleu);
imageJPEG($image);
?>
```

Figure 13.23 : *Image tournée*

Taille d'une image

Il existe deux fonctions permettant de récupérer la largeur et la hauteur d'une image : ce sont les fonctions `imageSX()` et `imageSY()`. Elles prennent toutes les deux comme paramètre l'identifiant d'une image :

imageSX()

Retourne la largeur de l'image.

Syntaxe	`int imageSX(resource $image)`

`$image`	Identifiant de l'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`retour`	Largeur de l'image (en pixels).

imageSY()

Retourne la hauteur de l'image.

Syntaxe	`int imageSX(resource $image)`
`$image`	Identifiant de l'image tel que retourné par la fonction `imageCreate()` et la famille de fonctions `imageCreateFromXX()`.
`retour`	Hauteur de l'image (en pixels).

Un exemple : l'histogramme

Nous allons effectuer, étape par étape, un script capable de dessiner dynamiquement un histogramme affichant la consommation de différents alcools (en litres) par Français.

Étape 1

À cette étape, nous allons simplement définir une image de fond :

Listing 13.23 : gd_histo1.php

```
<?php
  header("Content-type: image/png");
  $image = imageCreateFromJPEG("fondhisto.jpg");
  imagePNG($image);
  imageDestroy($image);
?>
```

Ici, nous récupérons simplement l'image `fondhisto.jpg`, qui sera l'image de fond de l'histogramme.

Figure 13.24 : *Étape 1*

Étape 2

Nous allons maintenant ajouter un titre :

Listing 13.24 : gd_histo2.php

```
<?php
  header("Content-type: image/png");
  $image     = imageCreateFromJPEG("fondhisto.jpg");
  $largeur   = imageSX($image);
  $hauteur   = imageSY($image);
  $fontTitre = 5;
  $rouge     = imageColorAllocate($image, 255, 0, 0);
  $titre     = "Consommation alcool";
  imageString($image, $fontTitre,
           ($largeur-ImageFontWidth($fontTitre)*strlen("$titre"))/2,
           0, $titre, $rouge);
  imagePNG($image);
  imageDestroy($image);
?>
```

À partir du script de l'étape 1, nous récupérons la largeur et la hauteur de l'image, puis nous définissons une police d'écriture parmi les cinq mises à disposition par PHP. Nous allouons une nouvelle couleur, puis nous prenons soin d'afficher le texte centré sur l'image.

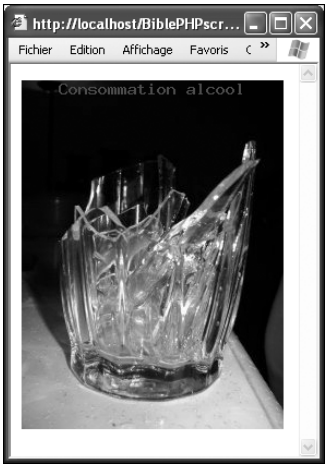


Figure 13.25 : *Étape 2*

Étape 3

Maintenant, nous allons tracer les rectangles. Nous avons choisi de passer les données dans l'adresse de la forme :

```
c13-histo.php?valeur1;libelle1;valeur2;libelle2;valeur3;libelle3…
```

Listing 13.25 : gd_histo3.php

```
<?php
  header("Content-type: image/png");
  $image         = imageCreateFromJPEG("fondhisto.jpg");
  $largeur       = imageSX($image);
  $hauteur       = imageSY($image);
  $fontTitre     = 5;
  $rouge         = imageColorAllocate($image, 255, 0, 0);
  $couleur_barre = imageColorAllocate($image,0,255,0);
  $titre         = "Consommation alcool";
  imageString($image, $fontTitre,
         ($largeur-ImageFontWidth($fontTitre)*strlen("$titre"))/2,
         0, $titre, $rouge);

  $valeurs      = explode(";", $argv[0]);
  $largeurBarre = (int)(($largeur)/(1.5*sizeof($valeurs)/2+0.5));
  $max = 0;
  for ($i=0; $i<sizeof($valeurs)/2; $i++) {
    if ($valeurs[$i*2]>$max) $max = $valeurs[$i*2];
  }
  for ($i=0;$i<sizeof($valeurs)/2;$i++) {
      $x = (int)($largeurBarre*(0.5+$i*1.5));
      $hauteurBarre   = (int)(($valeurs[$i*2]*($hauteur-40))/$max);
      $imageBarre     = imageCreate($largeurBarre, $hauteurBarre);
      $fondTemporaire = imageColorAllocate($imageBarre, 255, 255, 255);
      imageCopyMerge($image, $imageBarre, $x,
                     $hauteur-15-$hauteurBarre,
                     0, 0, $largeurBarre, $hauteurBarre, 70);
      imageDestroy($imageBarre);
  }
  imagePNG($image);
  imageDestroy($fondBarre);
  imageDestroy($image);
?>
```

Après avoir récupéré les valeurs passées de l'URL dans un tableau, nous calculons la largeur en pixels que devra avoir chacune des barres. Ensuite, afin d'exploiter au mieux l'espace vertical, il faut rechercher la valeur maximale parmi celles fournies.

La seconde boucle `for` dessine un à un les rectangles de l'histogramme.

Nous avons choisi ici de définir une image pour chacun des rectangles, afin de pouvoir faire un effet de transparence grâce à `imageCopyMerge()`.

Figure 13.26 :
Étape 3
c13-histo3.php?60;vin;39.6;biere;2.4;spiritueux;8.6;cidre

Étape 4

Pour ce script final, nous utiliserons une petite image PNG qui est un dégradé allant du vert au transparent.

Par rapport au script précédent, l'image des barres est passée en 'TrueColor'.

Listing 13.26 : gd_histo4.php

```
<?php
  header("Content-type: image/png");
  $image     = imageCreateFromJPEG("fondhisto.jpg");
  $largeur   = imageSX($image);
  $hauteur   = imageSY($image);
  $fontTitre = 5;
  $rouge     = imageColorAllocate($image, 255, 0, 0);
  $couleur_barre = imageColorAllocate($image,0,255,0);
  $titre     = "Consommation alcool";
  imageString($image, $fontTitre,
          ($largeur-ImageFontWidth($fontTitre)*strlen("$titre"))/2,
          0, $titre, $rouge);

  $valeurs = explode(";", $argv[0]);
  $largeurBarre = (int)(($largeur)/(1.5*sizeof($valeurs)/2+0.5));
  $max = 0;
  for ($i=0; $i<sizeof($valeurs)/2; $i++) {
    if ($valeurs[$i*2]>$max) $max = $valeurs[$i*2];
  }

  $fondBarre = imageCreateFromPNG("barhisto.png");

  for ($i=0;$i<sizeof($valeurs)/2;$i++) {
      $x            = (int)($largeurBarre*(0.5+$i*1.5));
      $hauteurBarre = (int)(($valeurs[$i*2]*($hauteur-40))/$max);
      $imageBarre   = imageCreateTrueColor($largeurBarre, $hauteurBarre);
```

```
        imageCopy($imageBarre, $image, 0, 0, $x,
            $hauteur-15-$hauteurBarre, $largeurBarre, $hauteurBarre);
        imageAlphaBlending($imageBarre, true);
        imageCopyResampled($imageBarre, $fondBarre, 0, 0,
            0, 0, $largeurBarre, $hauteurBarre, imageSX($fondBarre),
            imageSY($fondBarre));
        imageCopyMerge($image, $imageBarre, $x, $hauteur-15-$hauteurBarre,
            0, 0, $largeurBarre, $hauteurBarre, 90);
        imageDestroy($imageBarre);
        imageStringUp($image, 3, $x, $hauteur-17, $valeurs[$i*2+1],
            urldecode($rouge));
    }
    imagePNG($image);
    imageDestroy($fondBarre);
    imageDestroy($image);
?>
```

Pour rendre l'effet souhaité, il va nous falloir agrandir l'image qui nous servira à dessiner les rectangles, puis copier cette image sur l'image de fond. Il faudra, en outre, préserver l'effet de transparence défini dans l'image servant pour les rectangles.

Tout d'abord, nous avons créé une image 'TrueColor' puis recopié la partie de l'image de fond afin de pouvoir recréer par la suite l'effet de transparence.

Ensuite, nous devons préciser l'utilisation des couches alpha, puis copier sur cette image un agrandissement du fond des rectangles.

Il ne reste plus qu'à coller cette image sur l'image de fond, en utilisant éventuellement, comme ici, un autre effet de transparence.

Nous avons également écrit verticalement les libellés des bâtons.

Au final, voici ce que l'on obtient pour l'appel suivant :

```
gd_histo4.php?60;vin;39.6;biere;2.4;spiritueux;8.6;cidre
```

Figure 13.27 : *Étape 4*
c13-histo4.php?60;vin;39.6;biere;2.4;spiritueux;8.6;cidre

Récupérer des informations sur un fichier image

Il existe d'autres fonctions ne faisant pas partie de la bibliothèque GD, mais permettant de récupérer des informations sur les fichiers images.

C'est tout d'abord la fonction `getImageSize()`.

---

getImageSize()

Retourne la taille et le type d'une image.

Syntaxe	`array getImageSize(string $nomFichier [, array $info])`
`$nomFichier`	Nom du fichier image.
`retour`	Tableau indexé contenant : à l'index 0, la largeur de l'image (en pixels). à l'index 1, la hauteur de l'image (en pixels). à l'index 2, le code du format de l'image (1 = GIF, 2 = JPG, 3 = PNG, 4 = SWF, 5 = PSD, 6 = BMP, 7 = TIFF (ordre des octets intel), 8 = TIFF (ordre des octets motorola), 9 = JPC, 10 = JP2, 11 = JPX). à l'index 3, une chaîne de caractères `width=".." height="..."` pouvant être directement intégrée dans une balise `<img>`.

Récupérer des informations EXIF sur un fichier image

D'autres fonctions non originaires de la bibliothèque GD mais EXIF permettent de récupérer des informations sur les fichiers images. Les informations EXIF sont généralement inscrites dans les images prises par des appareils photo numériques. On peut retrouver dans ces informations la résolution, l'ouverture, la focale… Ces données varient d'un constructeur à l'autre.

Installation

Sous Windows

Avec l'archive du PHP Group

Vous devrez vous assurer d'avoir le fichier *php_exif.dll* (livré dans l'archive PHP distribuée par le PHP Group) dans votre répertoire contenant les extensions PHP, et ajouter ou décommenter une ligne

```
extension=php_exif.dll
```

Avec EasyPHP

Avec EasyPHP, le support d'EXIF est activé automatiquement.

Sous Linux

Vous devrez simplement recompiler PHP avec l'option `--enable-exif`.

Vérification

Vous pouvez vérifier le succès de l'opération avec un script contenant simplement la ligne `<?php phpinfo(); ?>` qui devra afficher :

exif	
EXIF Support	enabled
EXIF Version	1.2
Supported EXIF Version	02100
Supported filetypes	JPEG,TIFF

Figure 13.28 : *phpinfo()*

Utilisation

exif_imageType

Permet de déterminer le type d'une image.

Syntaxe `int exif_imageType(string $nomFichier)`

`$nomFichier` Nom et chemin du fichier image à étudier.

`retour` Type de l'image.

`IMAGETYPE_GIF` (= 1) au format GIF.
`IMAGETYPE_JPG` (= 2) au format JPEG.
`IMAGETYPE_PNG` (= 3) au format PNG.
`IMAGETYPE_SWF` (= 4) au format SWF.
`IMAGETYPE_PSD` (= 5) au format PSD.
`IMAGETYPE_BMP` (= 6) au format BMP.
`IMAGETYPE_TIFF_II` (= 7) au format TIFF II.
`IMAGETYPE_TIFF_MM` (= 8) au format TIFF MM.
`IMAGETYPE_JPC` (= 9) au format JPC.
`IMAGETYPE_JP2` (= 10) au format JP2.
`IMAGETYPE_JPX` (= 11) au format JPX.

exif_read_data()

Cette fonction permet de lire l'en-tête EXIF d'une image JPEG ou TIFF. (Disponible depuis PHP 4.2.0).

Syntaxe
```
array exif_read_data(string $nomFichier [, string
$informations [, boolean $tableaux [, boolean $thumbnail]]])
```

`$nomFichier` Nom du fichier image (ne peut pas être une URL).

`$informations`	Liste des informations (séparées par des virgules) que vous souhaitez extraire.
	`FILE` : Des informations sur le fichier de l'image (voir exemple)
	`COMPUTED` : Des informations sur la taille de l'image entre autre.
	`ANY_TAG`
	`IFD0`
	`THUMBNAIL` : Toutes les informations sur l'aperçu d'image.
	`COMMENT` : Eventuel commentaire sur la photo
	`EXIF` : Données concernant essentiellement les photos numériques.

`$tableaux`	`TRUE` si vous souhaitez que les informations soient regroupées par thème dans des tableaux, `FALSE` (par défaut) sinon. Les informations portant des noms pouvant créer des conflits seront quoi qu'il arrive regroupées dans des tableaux.
`$thumbnail`	`TRUE` si la fonction doit lire le *thumbnail* (vignette), `FALSE` sinon.
`retour`	Tableau associatif.

Voici un exemple qui permet de lire tout l'en-tête hormis l'aperçu :

Listing 13.27 : exif1.php

```
<?php
$fichier="adrien.jpg";
$exifInfos = read_exif_data($fichier);
echo $fichier.":<br />\n";
foreach($exifInfos as $cle=>$info) {
    if($cle == "Thumbnail"){
        $file = fopen("thumbnail","wb");
        fwrite($file, $info);
        fclose($file);
        echo "<img src='thumbnail' /><br />\n";
    } else {
        echo "$cle: $info<br />\n";
    }
}
?>
```

Figure 13.29 :
Photo de test : adrien.jpg

Voici ce qui est retourné dans le cas de cette photo (l'imagette a été retirée) :

```
adrien.jpg:
FILE.FileName: adrien.jpg
```

```
FILE.FileDateTime: 1027957704
FILE.FileSize: 561805
FILE.FileType: 2
FILE.MimeType: image/jpeg
FILE.SectionsFound: ANY_TAG, IFD0, THUMBNAIL, EXIF, MAKERNOTE
COMPUTED.html: width="1440" height="2160"
COMPUTED.Height: 2160
COMPUTED.Width: 1440
COMPUTED.IsColor: 1
COMPUTED.ByteOrderMotorola: 0
COMPUTED.CCDWidth: 15mm
COMPUTED.ExposureTime: 0.017 s (1/60)
COMPUTED.ApertureFNumber: f/4.5
COMPUTED.UserComment:
COMPUTED.UserCommentEncoding: UNDEFINED
COMPUTED.Thumbnail.FileType: 2
COMPUTED.Thumbnail.MimeType: image/jpeg
COMPUTED.Thumbnail.Height: 160
COMPUTED.Thumbnail.Width: 120
IFD0.Make: Canon
IFD0.Model: Canon EOS D30
IFD0.Orientation: 1
IFD0.XResolution: 180/1
IFD0.YResolution: 180/1
IFD0.ResolutionUnit: 2
IFD0.DateTime: 2001:12:25 06:35:11
IFD0.YCbCrPositioning: 1
IFD0.Exif_IFD_Pointer: 196
THUMBNAIL.Compression: 6
THUMBNAIL.XResolution: 180/1
THUMBNAIL.YResolution: 180/1
THUMBNAIL.ResolutionUnit: 2
THUMBNAIL.JPEGInterchangeFormat: 1244
THUMBNAIL.JPEGInterchangeFormatLength: 7537
<<ICI NOUS AURIONS L'IMAGETTE>>
EXIF.ISOSpeedRatings: 100
EXIF.ExifVersion: 0200
EXIF.DateTimeOriginal: 2001:12:25 06:35:11
EXIF.DateTimeDigitized: 2001:12:25 06:35:11
EXIF.ComponentsConfiguration: [1]
EXIF.ShutterSpeedValue: 387114/65536
EXIF.ApertureValue: 284416/65536
EXIF.ExposureBiasValue: 0/6
EXIF.FocalLength: 85/1
EXIF.UserComment:
EXIF.FlashPixVersion: 0100
EXIF.ColorSpace: 1
EXIF.ExifImageWidth: 1440
EXIF.ExifImageLength: 2160
EXIF.FocalPlaneXResolution: 1440000/595
EXIF.FocalPlaneYResolution: 2160000/892
EXIF.FocalPlaneResolutionUnit: 2
```

```
MAKERNOTE.ModeArray: Array
MAKERNOTE.UndefinedTag:0x0002: Array
MAKERNOTE.ImageInfo: Array
MAKERNOTE.ImageType: IMG:EOS D30 JPEG
MAKERNOTE.FirmwareVersion: Firmware Version 00.00
MAKERNOTE.ImageNumber: 1979721
MAKERNOTE.OwnerName:
MAKERNOTE.Camera: 422051855
MAKERNOTE.CustomFunctions: Array
```

Bien entendu, ces informations ne sont valables que pour cet appareil photo CANON EOS D30). Pour un autre appareil photo, les informations fournies seront probablement différentes, et cela sera encore plus vrai pour une autre marque d'appareil photo.

exif_thumbnail()

Retourne l'aperçu contenu dans l'en-tête EXIF s'il y en a un.

Syntaxe	`string exif_thumbnail(string $nomFichier[, int &$largeur[, &$hauteur]])`
`$nomFichier`	Nom du fichier dont vous souhaitez extraire l'aperçu.
`$largeur`	Référence vers la variable `$largeur` ; cette variable sera assignée par la valeur de la largeur de l'image.
`$hauteur`	Référence vers la variable `$hauteur` ; cette variable sera assignée par la valeur de la hauteur de l'image.
`retour`	Aperçu de l'image (*thumbnail*).

13.2. Les animations Flash

Comme vous le savez certainement, il est possible, grâce à la technologie Flash de Macromedia, de réaliser des animations multimédias interactives. Ces animations peuvent être intégrées dans une page web, pour peu que le navigateur intègre l'extension (plugin) ad hoc. Ces animations sont généralement construites une fois pour toutes, à partir d'un logiciel propriétaire. Mais, grâce aux bibliothèques dont dispose PHP, il est possible de générer ces animations au vol, ce qui permettra (même si ce n'est pas vraiment le cas dans les exemples suivants) de faire des animations qui varient en fonction du temps, de l'utilisateur, ou de tout autre paramètre dont le serveur aura connaissance.

Pour parvenir à ce résultat, PHP propose un bibliothèque appelée `Ming` qui, bien que considérée comme expérimentale, fonctionne déjà à merveille (il lui manque quand même encore quelques fonctionnalités).

Installation

La bibliothèque `Ming` est disponible aussi bien sous Windows que sous Linux.

Sous Windows

Avec l'archive du PHP Group

Vous devrez vous assurer d'avoir le fichier *php_ming.dll* (livré dans l'archive PHP distribuée par le PHP Group) dans votre répertoire contenant les extensions PHP, et ajouter ou décommenter une ligne

```
extension=php_ming.dll
```

Avec EasyPHP

Avec EasyPHP, le support de la bibliothèque `Ming` est activé automatiquement.

Sous Linux

Dans un premier temps, vous devrez vous procurer les sources de la bibliothèque. Celles-ci sont disponibles à l'adresse **http://ming.sourceforge.net /** (mais également sur le CD-ROM fourni).

Vous pourrez alors (par exemple, sous le répertoire */usr/local/src/lib)* taper les commandes suivantes :

```
# gunzip ming-0.2a.tgz
# tar xvf ming-0.2a.tar
```

afin de décompresser les sources.

```
# cd ming-0.2a
# make
```

Vous voilà maintenant avec un fichier *libming.so.*

Copiez *libming.so* sous */usr/local/lib.*

```
# cp libming.so /usr/local/lib/.
```

Copiez *ming.h* sous */usr/local/include.*

```
# cp ming.h /usr/local/include/.
```

Puis, vous devez recompiler PHP avec l'option `--with-ming=/usr/local.`

Problème d'installation et Apache
Si la librairie est mal installée, vous aurez certainement des messages du genre "child pid ... exit signal segmentation fault" dans le fichier de trace des erreurs d'Apache.

Vérification

Vous pouvez vous assurer du bon déroulement de l'opération d'installation en exécutant un simple script `<?php phpinfo(); ?>` devant afficher :

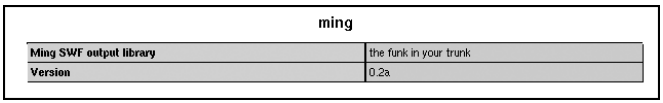


Figure 13.30 : *phpinfo()*

Utilisation

Comme tout type d'animation, les animations Flash sont composées d'une série de vues (en anglais, frames), et c'est la succession de ces vues qui donne la sensation de mouvement.

Chacune de ces vues contient un dessin "complexe", réalisé par la superposition de couches contenant des dessins "élémentaires". On pourra ainsi avoir un arrière-plan, un plan principal, un premier plan, etc.

Par défaut, chaque nouvelle vue contiendra l'ensemble des dessins de la vue précédente (ce qui évite, par exemple, de toujours redéfinir l'arrière-plan). Libre ensuite au réalisateur d'ajouter, de supprimer ou de déplacer, entre deux vues, certains éléments pour que, au final, l'animation donne une sensation de mouvement.

Figure 13.31 : *Exemple d'une courte animation*

Notez toutefois que le déroulement de l'animation n'est pas nécessairement linéaire. Comme l'on peut introduire de l'interactivité, il est à tout moment possible de revenir à une vue précédente ou de sauter à une tout autre vue.

Le schéma de base

L'animation Flash pourra, au choix, être directement envoyée au navigateur ou stockée dans un fichier.

Ainsi, la création d'un animation compte trois actions principales :

- Création d'un objet *Animation* (ou Film) par instanciation de l'objet *SWFMovie()* (qui ne nécessite aucun paramètre).
- Création des objets qui vont constituer l'animation, et association à l'objet *Animation*.
- Appel à la méthode `output()` de l'objet *Animation* pour un envoi direct au navigateur, ou appel à la méthode `save($nomFichier)` de l'objet *Animation* pour une sauvegarde de l'animation dans un fichier.

SWFMovie

Instancie un objet animation Flash.

Syntaxe	`SWFMovie SWFMovie(int $version)`
`$version`	Numéro de version de Flash. Par défaut 4.
`retour`	Objet *SWFMovie*.

SWFMovie->output()

Envoie le code de l'animation directement au navigateur.

Syntaxe	`int output(void)`
`retour`	La signification du code retour n'est pas connue.

SWFMovie->save()

Sauve le code de l'animation dans un fichier.

Syntaxe	`int save(string $nomFichier)`
`$nomFichier`	Nom du fichier de sauvegarde de l'animation.
`retour`	La signification du code retour n'est pas connue. En cas d'erreur, le script lève une erreur de type "fatale" et s'arrête.

Envoi direct au navigateur
Si vous décidez d'envoyer directement le contenu de l'animation Flash au navigateur, vous devez faire un appel à la fonction `header("Content-type: application/x-shockwave-flash")` *avant tout appel à la méthode* `output()`.
Vous pouvez choisir ici entre deux règles de codage :
Vous pouvez commencer directement le script par l'appel à la fonction `header()`. *Ainsi, à la lecture du code source, l'objectif du script sera de générer une animation Flash. Cependant, la contrepartie est que tout éventuel message d'erreur levé par les fonctions appelées après* `header()` *sera invisible, car interprété comme du code Flash.*
Vous pouvez également placer l'instruction `header()` *juste avant l'appel à la méthode* `output()`.

Au choix :

```
<?php
    header("Content-type: application/x-shockwave-flash");

    $anim = new Movie();
    // Construction proprement dite de l'animation
```

```
    $anim->output();
?>
```

ou

```
<?php
    $anim = new Movie();
    // Construction proprement dite de l'animation

    header("Content-type: application/x-shockwave-flash");
    $anim->output();
?>
```

Présentation des vues

Une fois la création de l'animation initialisée avec l'instanciation de l'objet *SWFMovie*, le générateur est prêt à composer la première vue. À partir de là, toutes les opérations s'appliqueront donc à la première vue. Une fois cette vue réalisée (cf. composition des vues), il faut appeler la méthode `nextFrame()` de l'objet *SWFMovie* pour valider la vue qui vient d'être créée. Cette fonction ne nécessite aucun paramètre, et ne retourne rien ; mais elle a également pour effet d'incrémenter le compteur de vues. Ainsi, toutes les opérations suivantes s'appliqueront non pas à la première, mais à la seconde vue, et ainsi de suite.

SWFMovie()->nextFrame()

Valide la vue en cours et passe à la création de la suivante.

Syntaxe `void nextFrame(void)`

Composition des vues

Chaque dessin qui compose une vue est réalisé à partir d'objets élémentaires (objets au sens informatique du terme). Parmi ces objets nous trouvons :

- Les textes ;
- Les formes "complexes" (basées sur des lignes, courbes, etc.) ;
- Les boutons ;
- Les zones de saisie de texte.

Certains de ces objets vont utiliser d'autres objets tels que :

- Les polices de caractères ;
- Les images ;
- Les dégradés ;

- Les remplissages.

De même, il existe d'autres objets de plus haut niveau comme :

- Les morphing ;
- Les sous-animations.

Une fois instancié, l'objet pourra alors être associé à la vue en cours par un simple appel à la méthode `add()` de l'objet *SWFMovie*.

SWFMovie->add()

Ajoute un objet à la vue courante de l'animation. Les objets graphiques s'empilent sur la couche supérieure aux précédents. Autrement dit, le dernier objet ajouté se retrouve en premier plan. Par la suite, nous verrons qu'il est également possible d'ajouter des objets qui ne sont pas des objets graphiques.

Syntaxe	`SWFDisplayItem add(mixed $objet)`
`$objet`	Objet à ajouter à la "sous-animation". Ce peut être un objet *SWFText*, *SWFTextField*, *SWFShape*, *SWFMorph*, *SWFButton*, *SWFAction* ou *SWFSprite*.
`retour`	Un objet *SWFDisplayItem* uniquement dans le cas des objets graphiques. Comme nous le verrons par la suite, cela permet de placer, transformer, etc. l'objet.

Il est possible d'ajouter plusieurs fois le même objet (la même représentation graphique) et d'obtenir ainsi plusieurs instances de *SWFDisplayItem*. Ce qui permet à chacun de ces objets d'avoir sa propre vie.

Les propriétés de l'animation peuvent être définies par appel aux méthodes suivantes :

SWFMovie->setDimension()

Détermine les dimensions pour lesquelles l'animation a été conçue. Il s'agit des dimensions "idéales", mais cela n'empêchera pas l'animation d'être affichée dans une taille supérieure si on le lui demande (via les attributs "`width`" et "`height`" de la balise HTML par exemple).

Syntaxe	`void setDimension(int $largeur, int $hauteur)`
`$largeur`	Largeur idéale de l'animation, en pixels.
`$hauteur`	Hauteur idéale de l'animation, en pixels.

SWFMovie->setBackground()

Définit la couleur de fond de l'animation.

Syntaxe	`void setBackground(int $rouge, int $vert, int $bleu)`
`$rouge`	Composante rouge de la couleur de fond. Valeur comprise entre 0 et 255.
`$vert`	Composante verte de la couleur de fond. Valeur comprise entre 0 et 255.
`$bleu`	Composante bleue de la couleur de fond. Valeur comprise entre 0 et 255.

SWFMovie->setRate()

Définit la vitesse de défilement de l'animation. La vitesse de défilement est toutefois dépendante de la vitesse d'affichage et de traitement des instructions Flash. De ce fait, la vitesse observée peut être plus lente que prévue, ou bien quelques vues peuvent ne pas être affichées.

Syntaxe `void setRate(double $vitesseDefilement)`

`$vitesseDefilement` Vitesse de défilement de l'animation en nombre d'images par seconde.

Listing 13.28 : ming_01.php

```
<?php

    // Cette animation Flash ne fait rien
    // du tout
    //
    // Ce script présente simplement
    // la structure classique d'une animation Flash

    // Instanciation de l'objet animation

    $anim = new SWFMovie();

    // Définition des principaux paramètres
    // - Dimension
    // - Couleur de fond
    // - Vitesse de défilement

    $anim->setDimension(100, 100);
    $anim->setBackground(255, 255, 255);
    $anim->setRate(1);

    //
    // Ici, on pourrait trouver
    // une série d'instanciations d'objets
    // disons $objet1, $objet2, etc...
    //
```

```
    // Ensuite, on peut
    // composer la vue 1

    $anim->add($objet1);
    $anim->add($objet2);

    // Valider la vue 1

    $anim->nextFrame();

    // Pour éventuellement passer
    // à la composition de la vue 2.
    // Où on pourra ajouter d'autres objets

    $anim->add($objet3);
    $anim->nextFrame();


    // Et enfin, on pourra envoyer le flux
    // directement au navigateur
    // après l'avoir prévenu que ce qui suit
    // est une animation Flash
    header("Content-type: application/x-shockwave-flash");
    $anim->output();
?>
```

L'objet texte (Text)

Les textes s'instancient par un appel au constructeur `SWFText()`.

SWFText

Objet texte.

Syntaxe `SWFText SWFText(void)`

Cet objet propose une série de méthodes permettant de définir la police qui doit être utilisée, l'endroit où doit être positionné le texte et, évidemment, le texte qui doit être affiché.

SWFText->setFont()

Précise quelle police de caractères doit être utilisée pour le texte qui vient.

Syntaxe `void setFont(SWFFont $police)`

`$police` Objet *SWFFont* (voir chapitre suivant).

SWFText->setHeight()

Précise quelle hauteur la police de caractères doit prendre pour le texte qui vient.

Syntaxe `void setHeight(int $hauteur)`

`$hauteur`	Hauteur de la police de caractères. L'unité de mesure est celle de l'animation (autrement dit en pixels si les dimensions "préconisées" pour l'animation sont respectées).

SWFText->setColor()

Précise la couleur à utiliser pour le texte qui vient. Tant qu'aucun appel à cette méthode n'est fait, le texte est transparent.

Syntaxe `void setColor(int $rouge, int $vert, int $bleu, [$alpha])`

`$rouge`	Composante rouge de la couleur du texte. Valeur comprise entre 0 et 255.
`$vert`	Composante verte de la couleur du texte. Valeur comprise entre 0 et 255.
`$bleu`	Composante bleue de la couleur du texte. Valeur comprise entre 0 et 255.
`$alpha`	Composante Alpha (transparence) de la couleur du texte. Valeur optionnelle, comprise entre 0 (couleur totalement transparente) et 255 (couleur totalement opaque).

SWFText->moveTo()

Positionne le texte qui vient.

Syntaxe `void moveTo(int $x, int $y)`

`$x`	Coordonnée x du prochain texte (l'axe des abscisses est dirigé de la gauche vers la droite).
`$y`	Coordonnée y du prochain texte (l'axe des ordonnées est dirigé du haut vers le bas).

SWFText->addString()

Complète le texte à afficher. Le texte sera affiché avec les propriétés définies par les méthodes vues précédemment (et appelées avant `addString()`).

Syntaxe `void addString(string $texte)`

`$texte`	Texte à ajouter à l'objet *SWFText*.


REMARQUE

En cours de développement
Il existe une méthode `setSpacing()` *censée modifier l'espace entre les caractères, mais dans la version testée, cette méthode ne fonctionne pas.*

Voici ce que, déjà, vous pouvez faire :

Listing 13.29 : ming_text01.php

```
<?php

    // Initialisation d'une nouvelle animation
    // à un rythme très léger 1 image/seconde
    // pour bien détailler la scène
    $anim = new SWFMovie();
    $anim->setDimension(100, 100);
    $anim->setBackground(255, 255, 255);
    $anim->setRate(1);

    // Instanciation d'un nouvel objet police
    // Avant d'executer ce script, pensez
    // à vérifier que vous possédez bien
    // cette police dans votre environnement
    // et que le chemin vers le fichier .fdb
    // est bien conforme à votre environnement.
    // Sinon, vous risquez le crash de votre navigateur

    $police = new SWFFont("ming_polices/arial.fdb");


    // Création de l'objet texte1
    $texte1 = new SWFText();

    // Opération indispensable
    // préciser la police de caractère du texte
    $texte1->setFont($police);

    // Selection de la couleur du texte
    $texte1->setColor(0, 0, 0);

    // Positionnement du texte
    $texte1->moveTo(10, 10);
    $texte1->setHeight(10);

    $texte1->addString("Faire du Flash ...");

    // Création de l'objet texte2
    $texte2 = new SWFText();
    $texte2->setFont($police);
    $texte2->setColor(0, 0, 0);
    $texte2->moveTo(10, 30);
    $texte2->setHeight(20);
```

```
$texte2->addString("... avec PHP ?");
$texte2->setHeight(20);

// Création de l'objet texte3
// centré
$texte3 = new SWFText();
$texte3->setFont($police);
$texte3->setColor(0, 0, 0);
$chaine="Finallement, c'est ...";
$texte3->setHeight(5);
$texte3->moveTo(50-$texte3->getWidth($chaine)/2, 40);
$texte3->addString($chaine);


// Création de l'objet texte4
// avec des lettres de plus en plus grosses
$texte4 = new SWFText();
$texte4->setFont($police);
$texte4->setColor(255, 0, 0);
$texte4->moveTo(10, 70);
$texte4->setHeight(10);
$texte4->addString("Fa");
$texte4->setHeight(20);
$texte4->addString("ci");
$texte4->setHeight(30);
$texte4->addString("le !");

// La première vue ne contiendra
// que le premier texte
$anim->add($texte1);
$anim->nextFrame();

// Puis la seconde vue sera enrichie
// d'un second texte
$anim->add($texte2);
$anim->nextFrame();

// puis d'un troisième
$anim->add($texte3);
$anim->nextFrame();

// et le texte sera complet
// dans la dernière vue
$anim->add($texte4);
// Bien qu'il s'agisse de la dernière vue
// il est indispensable de faire appel
// à nextFrame();
$anim->nextFrame();


header("Content-type: application/x-shockwave-flash");
$anim->output();
?>
```

Préciser une police de caractères
Il n'y a pas de police par défaut. Avant d'ajouter un texte, il faut obligatoirement indiquer quelle police devra être utilisée.

Couleur par défaut
Le texte étant par défaut transparent, si vous ne voyez rien à l'écran, c'est peut-être tout simplement que vous avez oublié de faire un appel à la méthode `setColor()`.

L'objet champ texte (TextField)

Les champs textes s'instancient par un appel au constructeur `SWFTextField()`.

SWFTextField

Objet champ texte.

Syntaxe `SWFTextField SWFTextField([int $options])`

`$options` Paramètre optionnel précisant le comportement et le rendu de l'objet. Cet argument peut prendre une ou des valeurs (en les combinant par l'opérateur logique `OU "|"`) parmi :

`SWFTEXTFIELD_NOEDIT` : champ texte non éditable.
`SWFTEXTFIELD_PASSORD` : champ mot de passe (les caractères sont remplacés à l'affichage par des `*`).
`SWFTEXTFIELD_DRAWBOX` : trace le contour du champ texte.
`SWFTEXTFIELD_MULTILINE` : crée un champ texte multi-ligne (selon les dimensions du champ texte).
`SWFTEXTFIELD_WORDWRAP` : force le retour à la ligne des mots.
`SWFTEXTFIELD_NOSELECT` : champ texte que l'on ne peut pas sélectionner (valeur par défaut).

Cet objet propose une série de méthodes permettant de définir la police qui doit être utilisée, la taille que doit avoir le champ texte, etc.

SWFTextField->setBounds()

Précise la largeur et la hauteur du champ texte.

Syntaxe `void setBounds(int $largeur, int $hauteur)`

`$largeur` Largeur du champ texte.

`$hauteur` Hauteur du champ texte.

SWFTextField->setColor()

Précise la couleur à utiliser pour le texte qui sera ajouté au champ texte. Tant qu'aucun appel à cette méthode n'est fait, le texte est noir.

Syntaxe	`void setColor(int $rouge, int $vert, int $bleu, [$alpha])`
`$rouge`	Composante rouge de la couleur du texte. Valeur comprise entre 0 et 255.
`$vert`	Composante verte de la couleur du texte. Valeur comprise entre 0 et 255.
`$bleu`	Composante bleue de la couleur du texte. Valeur comprise entre 0 et 255.
`$alpha`	Composante Alpha (transparence) de la couleur du texte. Valeur optionnelle comprise entre 0 (couleur totalement transparente) et 255 (couleur totalement opaque).

SWFTextField->setFont()

Précise quelle police de caractères doit être utilisée pour le texte du champ texte.

Syntaxe	`void setFont(SWFFont $police)`
`$police`	Objet *SWFFont* (voir chapitre suivant).

SWFTextField->setHeight()

Précise quelle hauteur de police de caractères doit être utilisée pour le texte du champ texte.

Syntaxe	`void setHeight(int $hauteur)`
`$hauteur`	Hauteur de la police de caractères. L'unité de mesure est celle de l'animation (autrement dit en pixels si les dimensions "préconisées" pour l'animation sont respectées).

SWFTextField->align()

Précise l'alignement du texte dans le champ texte.

Syntaxe	`void align(int $alignement)`
`$alignement`	`$alignement` peut prendre une valeur parmi : `SWFTEXTFIELD_ALIGN_LEFT` pour un alignement à gauche. `SWFTEXTFIELD_ALIGN_RIGHT` pour un alignement à droite. `SWFTEXTFIELD_ALIGN_CENTER` pour un texte centré. `SWFTEXTFIELD_ALIGN_JUSTIFY` pour un texte justifié.

SWFTextField->setLeftMargin()

Précise la largeur de la marge gauche (espace séparant le texte du bord gauche du champ texte).

Syntaxe	`void setLeftMargin(int $largeur)`
`$largeur`	Largeur de la marge.

SWFTextField->setRightMargin()

Précise la largeur de la marge droite (espace séparant le texte du bord droit du champ texte).

Syntaxe	`void setRightMargin(int $largeur)`
`$largeur`	Largeur de la marge.

SWFTextField->setMargins()

Précise la largeur des marges gauche (espace séparant le texte du bord gauche du champ texte) et droite (espace séparant le texte du bord droit du champ texte).

Syntaxe	`void setMargins(int $largeurGauche, int $largeurDroite)`
`$largeurGauche`	Largeur de la marge gauche.
`$largeurDroite`	Largeur de la marge droite.

SWFTextField->setIndentation()

Précise la largeur du retrait à droite de la première ligne du champ texte.

Syntaxe	`void setIndentation(int $largeur)`
`$largeur`	Largeur du retrait à droite.

SWFTextField->setLineSpacing()

Précise l'espacement entre deux lignes de texte du champ texte.

Syntaxe	`void setLigneSpacing(int $hauteur)`
`$hauteur`	Espacement entre deux lignes.

SWFTextField->addString()

Précise ou ajoute un texte au contenu du champ texte.

Syntaxe	`void addString(string $texte)`
`$texte`	Texte à ajouter au champ texte.

L'objet Police (Font)

Les polices de caractères s'instancient par un appel au constructeur `SWFFont`.

SWFFont

Objet police de caractères.

Syntaxe	`SWFFont SWFFont(string $fichierPolice)`
`$fichierPolice`	Chemin vers un fichier *.dbf* contenant les caractéristiques de la police de caractères.

Les polices
Il n'y a pas de police fournie par défaut avec la bibliothèque `Ming`. Il vous faudra donc soit en créer vous-même, soit en récupérer sur Internet. Par exemple à l'adresse : ***http://www.opaque.net/wiki/index.php?MingFonts***.

Il n'y a pas de méthode destinée à modifier les propriétés des polices. Mais il est en revanche possible d'en récupérer les propriétés avec :

SWFFont->getWidth()

Indique la largeur qu'occupera, dans cette police, le texte indiqué. Comme il s'agit là de la largeur dans la taille par défaut, il est préférable de faire appel à la méthode `getWidth()` de l'objet *SWFText*, et seulement après avoir appelé la méthode `setHeight()` associée.

Syntaxe	`int getWidth(string $texte)`
`$texte`	Texte dont on veut mesurer la largeur dans cette police.
`retour`	Largeur du texte dans cette police.

SWFFont->getAscent()

Retourne la distance qui sépare la ligne de base du haut de la lettre la plus haute dans cette police (distance représentée par un `a` dans le schéma suivant).

Syntaxe	`int getAscent(void)`
`retour`	Distance qui sépare la ligne de base du haut de la lettre la plus haute dans cette police.

SWFFont->getDescent()

Retourne la distance qui sépare la ligne de base du bas de la lettre descendant le plus bas dans cette police (distance représentée par un `d` dans le schéma suivant).

Syntaxe	`int getDescent(void)`
`retour`	Distance qui sépare la ligne de base du bas de la lettre descendant le plus bas dans cette police.

SWFFont->getLeading()

Retourne la distance conseillée entre deux lignes de texte successives.

Syntaxe	`int getLeading(void)`
`retour`	Distance comprise entre la ligne de base de la première ligne et le haut de la plus haute lettre dans cette police de la seconde ligne (distance représentée par un `l` dans le schéma suivant).

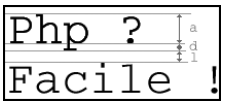


Figure 13.32 :
Les propriétés d'une police de caractères

L'objet forme "complexe" (Shape)

Les formes complexes s'instancient par l'appel au constructeur `SWFShape()`.

SWFShape

Objet forme "complexe".

Syntaxe	`SWFShape SWFShape()`

Cet objet propose de nombreuses méthodes permettant le tracé de lignes ou de courbes, le remplissage de formes, etc.

Lors du tracé des objets, il est important de noter que les axes des coordonnées sont dirigés de gauche à droite pour les x, et de haut en bas pour les y.

Les méthodes de tracé sont :

SWFShape->movePenTo()

Déplace le pointeur de tracé.

Syntaxe	`void movePenTo(int $x, int $y)`
`$x`	Nouvelle abscisse du pointeur de tracé.
`$y`	Nouvelle ordonnée du pointeur de tracé.

SWFShape->movePen()

Déplace le pointeur de tracé relativement à sa position courante.

Syntaxe	`void movePen(int $dx, int $dy)`
`$dx`	Déplacement selon l'axe des x du pointeur de tracé. La nouvelle abscisse absolue est alors `$xPointeur+$dx`.
`$dy`	Déplacement selon l'axe des y du pointeur de tracé. La nouvelle ordonnée absolue est alors `$yPointeur+$dy`.

SWFShape->drawLineTo()

Trace une droite entre la position courante du pointeur et le point précisé. Le pointeur est ensuite déplacé aux coordonnées précisées.

Syntaxe	`void drawLineTo(int $x, int $y)`
`$x`	Abscisse du point d'arrivée de la ligne tracée.
`$y`	Ordonnée du point d'arrivée de la ligne tracée.

SWFShape->drawLine()

Trace une droite entre la position courante du pointeur et le point précisé. Le pointeur est ensuite déplacé aux coordonnées précisées. Les coordonnées du point d'arrivée sont ici relatives à la position du pointeur.

Syntaxe	`void drawLine (int $dx, int $dy)`
`$x`	Abscisse relative du point d'arrivée de la ligne tracée, soit l'abscisse absolue `$xPointeur+$dx`.
`$y`	Ordonnée relative du point d'arrivée de la ligne tracée, soit l'ordonnée absolue `$yPointeur+$dy`.

SWFShape->drawCurveTo()

Trace une courbe de Bézier quadratique, issue de la position du pointeur et aboutissant aux dernières coordonnées données en prenant appui sur le point indiqué par les premières coordonnées. Le pointeur prend ensuite la position du point d'arrivée.

Syntaxe	`void drawCurveTo(int $x1, int $y1, int $x2, int $y2)`
`$x1`	Abscisse du point d'appui de la courbe de Bézier.
`$y1`	Ordonnée du point d'appui de la courbe de Bézier.
`$x2`	Abscisse du point d'arrivée de la courbe de Bézier et future abscisse du pointeur.
`$y2`	Ordonnée du point d'arrivée de la courbe de Bézier et future ordonnée du pointeur.

SWFShape->drawCurve()

Trace une courbe de Bézier quadratique, issue de la position du pointeur et aboutissant aux dernières coordonnées données en prenant appui sur le point indiqué par les premières coordonnées. Le pointeur prend ensuite la position du point d'arrivée. Dans ce cas, les coordonnées sont relatives.

Syntaxe	`void drawCurve(int $dx1, int $dy1, int $dx2, int $dy2)`
`$dx1`	Abscisse relative au pointeur du point d'appui de la courbe de Bézier, soit l'abscisse absolue `$xPointeur+$dx1`.
`$dy1`	Ordonnée relative au pointeur du point d'appui de la courbe de Bézier, soit l'ordonnée absolue `$yPointeur+$dy1`.
`$dx2`	Abscisse relative au point d'appui du point d'arrivée de la courbe de Bézier et future abscisse du pointeur, soit l'abscisse absolue `$xPointeur+$dx1+$dx2`.

`$dy2`	Ordonnée relative au point d'appui du point d'arrivée de la courbe de Bézier et future ordonnée du pointeur, soit l'ordonnée absolue `$yPointeur+$dy1+$dy2`.

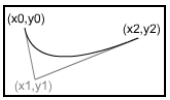


Figure 13.33 :
Exemple de courbe de Bézier quadratique

SWFShape->drawCubicTo()

Trace une courbe de Bézier cubique, issue de la position du pointeur et aboutissant aux dernières coordonnées données en prenant appui sur les deux points indiqués par les premières coordonnées. Le pointeur prend ensuite la position du point d'arrivée.

Syntaxe	`void drawCubicTo(int $x1, int $y1, int $x2, int $y2, int $x3, int $y3)`
`$x1`	Abscisse du 1er point d'appui de la courbe de Bézier.
`$y1`	Ordonnée du 1er point d'appui de la courbe de Bézier.
`$x2`	Abscisse du 2nd point d'appui de la courbe de Bézier.
`$y2`	Ordonnée du 2nd point d'appui de la courbe de Bézier.
`$x3`	Abscisse du point d'arrivée de la courbe de Bézier et future abscisse du pointeur.
`$y3`	Ordonnée du point d'arrivée de la courbe de Bézier et future ordonnée du pointeur.

SWFShape->drawCubic()

Trace une courbe de Bézier cubique, issue de la position du pointeur et aboutissant aux dernières coordonnées données en prenant appui sur les deux points indiqués par les premières coordonnées. Le pointeur prend ensuite la position du point d'arrivée. Dans ce cas, les coordonnées sont relatives.

Syntaxe	`void drawCubic(int $dx1, int $dy1, int $dx2, int $dy2, int $dx3, int $dy3)`
`$dx1`	Abscisse relative au pointeur du 1er point d'appui de la courbe de Bézier, soit l'abscisse absolue `$xPointeur+$dx1`.
`$dy1`	Ordonnée relative au pointeur du 1er point d'appui de la courbe de Bézier, soit l'ordonnée absolue `$yPointeur+$dy1`.
`$dx2`	Abscisse relative au 1er point d'appui du 2nd point d'appui de la courbe de Bézier, soit l'abscisse absolue `$xPointeur+$dx1+$dx2`.
`$dy2`	Ordonnée relative au 1er point d'appui du 2nd point d'arrivée de la courbe de Bézier, soit l'ordonnée absolue `$yPointeur+$dy1+$dy2`.

`$dx3`	Abscisse relative au 2nd point d'appui du point d'arrivée de la courbe de Bézier et future abscisse du pointeur, soit l'abscisse absolue `$xPointeur+$dx1+$dx2+$dx3`.
`$dy3`	Ordonnée relative au 2nd point d'appui du point d'arrivée de la courbe de Bézier et future ordonnée du pointeur, soit l'ordonnée absolue `$yPointeur+$dy1+$dy2+$dy3`.

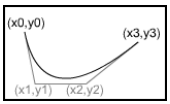


Figure 13.34 :
Exemple de courbe de Bézier cubique


REMARQUE

drawCubic() et drawCurve()
Pour obtenir des courbes de Bézier cubiques, il est également possible d'appeler les méthodes `drawCurve()` et `drawCurveTo()` avec les six mêmes paramètres que ceux de `drawCubic()` et `drawCubicTo()`.

SWFShape->drawGlyph()

Trace un caractère donné à la position du pointeur.

Syntaxe	`void drawGlyph(SWFFont $police, char $caractere)`
`$police`	Police d'écriture à utiliser.
`$caractere`	Caractère à écrire. Cette méthode ne permet de tracer qu'un caractère à la fois.


REMARQUE

Paramètres de tracé
Par défaut, pour toutes les méthodes présentées précédemment, la largeur du tracé est nulle. Vous ne verrez donc rien tant que vous ne ferez pas appel à la méthode `setLine()` ou aux fonctions de remplissage `setLeftFill()` et `setRightFill()`. Voir le chapitre dédié au tracé et remplissage.

Voici donc un exemple de ce qui peut être réalisé :

Listing 13.30 : ming_shape01.php

```
<?php

    // L'animation ne présentera
    // qu'une image par seconde
    // pour laisser le temps
    // d'apprécier la succession des vues
```

```
$anim = new SWFMovie();
$anim->setDimension(100, 100);
$anim->setBackground(0, 125, 255);
$anim->setRate(1);

// Création des différents objets
// qui seront utilisés dans
// l'animation

// La première forme sera une simple ligne
// partant du point (50, 95)
// et remontant de 45 unites
// Le tracé sera vert et d'1 unité de large.
$forme1 = new SWFShape();
$forme1->setLine(1, 0, 255, 0);
$forme1->movePenTo(50, 95);
$forme1->drawLine(0, -45);

// La seconde forme sera un carré
// de 20 unités de côté
// avec un tracé jaune d'1 unité de large
$forme2 = new SWFShape();
$forme2->setLine(1, 255, 255, 0);
$forme2->movePenTo(40, 50);
$forme2->drawLine(0, -20);
$forme2->drawLine(20, 0);
$forme2->drawLine(0, 20);
$forme2->drawLine(-20, 0);

// La troisième forme sera simplement
// le caractère *
// Mais pour cela, nous devons
// également créer un objet SWFFont
// Ceci nécessite donc que vous ayez
// un fichier Arial.fdb dans un sous-répertoire
// ming_polices.
// Si ce n'est pas le cas, modifier
// le nom de la police ou le chemin
// spécifié pour le faire coincider
// avec votre environnement.
// A défaut, mettez en commentaire
// la création de forme3 et son
// ajout à l'animation.
$forme3 = new SWFShape();
$font = new SWFFont("ming_polices/arial.fdb");
$forme3->setLine(1, 255, 255, 255);
$forme3->movePenTo(40, 70);
$forme3->drawGlyph($font, "*");

// Puis une courbe
$forme4 = new SWFShape();
$forme4->setLine(1, 0, 0, 0);
```

```
$forme4->movePenTo(50, 30);
$forme4->drawCubic(0, -50, 60, 60, -50, 0);

// et une autre
$forme5 = new SWFShape();
$forme5->setLine(1, 0, 0, 0);
$forme5->movePenTo(60, 40);
$forme5->drawCubic(50, 0, -60, 60, 0, -50);

// et une autre
$forme6 = new SWFShape();
$forme6->setLine(1, 0, 0, 0);
$forme6->movePenTo(50, 50);
$forme6->drawCubic(0, 50, -60, -60, 50, 0);

// et encore une autre
$forme7 = new SWFShape();
$forme7->setLine(1, 0, 0, 0);
$forme7->movePenTo(40, 40);
$forme7->drawCubic(-50, 0, 60, -60, 0, 50);

// Maintenant que l'on a tout les objets
// on passe à la création de l'abimation

// Dans la vue 1 n'apparait que la forme 1
$anim->add($forme1);
$anim->nextFrame();

// Puis dans la vue 2 vient s'ajouter la forme 2
$anim->add($forme2);
$anim->nextFrame();

// dans la vue 3.... la forme 3
$anim->add($forme3);
$anim->nextFrame();

// dans ma vue 4... tout le reste
$anim->add($forme4);
$anim->add($forme5);
$anim->add($forme6);
$anim->add($forme7);

// ATTENTION: Même s'il s'agit de la
// dernière vue, ne pas oublier
// l'appel à nextFrame();
$anim->nextFrame();

// Il est temps de voir le résultat
header("Content-type: application/x-shockwave-flash");
$anim->output();
?>
```


REMARQUE

Fonctions non disponibles
À ce jour, il n'est pas possible de tracer facilement des cercles ou arcs de cercles. Les fonctions `drawArc()` *et* `drawCircle()` *prévues à cet effet ne sont pas encore fonctionnelles.*

Positionnement et modification des objets

Comme cela a été évoqué précédemment, lorsque l'on appelle la méthode `add()` de l'objet *SWFMovie* pour ajouter un objet de type `SWFButton`, `SWFShape`, `SWFSprite` ou `SWFText`, un objet *SWFDisplayItem* est retourné.

Cet objet permet, grâce aux méthodes suivantes, de :

Donner un nom aux objets :

SWFDisplayItem->setName()

Donne un nom à l'objet afin de pouvoir l'identifier dans les scripts d'action (que nous verrons plus loin).

Syntaxe	`void setName(string $nom)`
`$nom`	Nom à donner à l'objet.

Modifier la position des objets :

SWFDisplayItem->setDepth()

Déplace l'objet dans un couche différente (pour le mettre devant ou derrière un ou plusieurs autres objets).

Syntaxe	`void setDepth(int $couche)`
`$couche`	Couche dans laquelle doit être déplacé l'objet.

SWFDisplayItem->moveTo()

Déplace l'origine de l'objet vers de nouvelles coordonnées.

Syntaxe	`void moveTo(int $x, int $y)`
`$x`	Nouvelle abscisse pour l'origine de l'objet.
`$y`	Nouvelle ordonnée pour l'origine de l'objet.

SWFDisplayItem->move()

Déplace l'origine de l'objet vers de nouvelles coordonnées relatives à sa position courante.

Syntaxe	`void move(int $dx, int $dy)`
`$dx`	Abscisse relative pour l'origine de l'objet. Soit l'abscisse absolue `$xCourant+$dx`.
`$dy`	Ordonnée relative pour l'origine de l'objet. Soit l'ordonnée absolue `$yCourant+$dy`.

SWFDisplayItem->rotateTo()

Fait pivoter l'objet par rapport à son angle d'origine.

Syntaxe	`void rotateTo(double $angle)`
`$angle`	Angle de rotation exprimé en degrés.

SWFDisplayItem->rotate()

Fait pivoter l'objet par rapport à son angle courant.

Syntaxe	`void rotate (double $dangle)`
`$angle`	Angle de rotation relatif à l'angle courant, exprimé en degrés.

D'altérer leur forme :

SWFDisplayItem->scaleTo()

Modifie la taille de l'objet par rapport à sa taille d'origine.

Syntaxe	`void scaleTo(double $coefX, double $coefY)`
`$coefX`	Coefficient multiplicateur à appliquer sur la largeur d'origine de l'objet.
`$coefY`	Coefficient multiplicateur à appliquer sur la hauteur d'origine de l'objet.

SWFDisplayItem->scale()

Modifie la taille de l'objet par rapport à sa taille courante.

Syntaxe	`void scale(double $coefX, double $coefY)`
`$coefX`	Coefficient multiplicateur à appliquer sur la largeur courante de l'objet.
`$coefY`	Coefficient multiplicateur à appliquer sur la hauteur courante de l'objet.

SWFDisplayItem->skewXTo()

Déforme l'objet par une inclinaison de l'axe x selon une pente donnée par rapport à son inclinaison d'origine.

Syntaxe	`void skewXTo(double $pente)`
`$pente`	Coefficient de pente, ou encore tangente de l'angle d'inclinaison.

SWFDisplayItem->skewX()

Déforme l'objet par une inclinaison de l'axe x selon une pente donnée par rapport à son inclinaison courante.

Syntaxe	`void skewX(double $pente)`
`$pente`	Coefficient de pente, ou encore tangente de l'angle d'inclinaison.

SWFDisplayItem->skewYTo()

Déforme l'objet par une inclinaison de l'axe y selon une pente donnée par rapport à son inclinaison d'origine.

Syntaxe	`void skewYTo(double $pente)`
`$pente`	Coefficient de pente, ou encore tangente de l'angle d'inclinaison.

SWFDisplayItem->skewY()

Déforme l'objet par une inclinaison de l'axe y selon une pente donnée par rapport à son inclinaison courante.

Syntaxe	`void skewY(double $pente)`
`$pente`	Coefficient de pente, ou encore tangente de l'angle d'inclinaison.

ou leur couleur :

SWFDisplayItem->addColor()

Ajoute (ou soustrait selon les signes) aux composantes de la couleur d'origine les composantes de la couleur donnée. Les valeurs obtenues seront évidemment bornées sur l'intervalle [0, 255].

Syntaxe	`void addColor(int $rouge, int $vert, int $bleu, [int $alpha])`
`$rouge`	Composante rouge (comprise entre -255 et 255) à ajouter à la couleur d'origine.
`$vert`	Composante verte (comprise entre -255 et 255) à ajouter à la couleur d'origine.
`$bleu`	Composante bleue (comprise entre -255 et 255) à ajouter à la couleur d'origine.
`$alpha`	Composante Alpha ou transparence (comprise entre -255 et 255) à ajouter à la couleur d'origine. Rappel : plus Alpha est proche de 0, plus l'objet est transparent ; plus il est proche de 255, plus l'objet est opaque.

SWFDisplayItem->multColor()

Multiplie les composantes de la couleur d'origine. Les valeurs obtenues seront évidemment bornées sur l'intervalle [0, 255].

Syntaxe	`void multColor(double $coefRouge, double $coefVert, double $coefBleu, [double $coefAlpha])`
`$coefRouge`	Coefficient multiplicateur pour la composante rouge (> = 0).
`$coefVert`	Coefficient multiplicateur pour la composante verte (> = 0).
`$coefBleu`	Coefficient multiplicateur pour la composante bleue (> = 0).
`$alpha`	Coefficient multiplicateur pour la composante Alpha ou transparence (> = 0). Rappel : plus Alpha est proche de 0, plus l'objet est transparent ; plus il est proche de 255, plus l'objet est opaque.

Axe de rotation
Les rotations s'effectuant par rapport à l'axe qui a servi d'origine lors de la création de l'objet, il est fortement conseillé de faire coïncider cet axe avec le centre de l'objet. L'erreur classique consiste à créer l'objet autour du point dans lequel il se trouve dans la première vue.

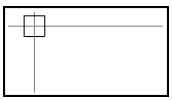


Figure 13.35 :
Exemple d'un objet créé avec le centre à l'origine

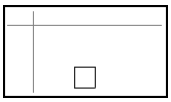


Figure 13.36 :
puis ayant subi une translation

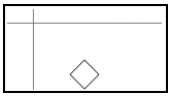


Figure 13.37 :
suivie d'une rotation de 45 °.

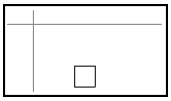


Figure 13.38 :
Exemple d'un objet créé à son emplacement dans la première vue

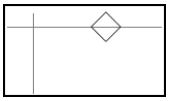


Figure 13.39 :
puis ayant subit une rotation de 45 °.

Suppression d'un objet

En revanche, pour supprimer un objet, on fera appel à la méthode `remove()` de l'objet *SWFMovie*.

SWFMovie->remove()

Supprime un objet de la vue.

Syntaxe `void remove(SWFDisplayItem $objetSWFDisplayItem)`

`$objetSWFDisplayItem` L'objet *SWFDisplayItem* qui avait été retourné par la méthode `add()` lorsque l'objet avait été ajouté à une précédente vue.

Dans l'exemple suivant, nous avons deux objets identiques, dont l'un est supprimé dans la seconde vue.

Listing 13.31 : ming_remove.php

```
<?php

    $dessinForme = new SWFShape();

    $dessinForme->setLine(1, 0, 0, 0);
    $dessinForme->movePenTo(-10, -10);
```

```
    $dessinForme->drawLine(20, 0);
    $dessinForme->drawLine(0, 20);
    $dessinForme->drawLine(-20, 0);
    $dessinForme->drawLine(0, -20);

    $anim = new SWFMovie();
    $anim->setDimension(100, 100);
    $anim->setBackground(255, 255, 255);
    $anim->setRate(1);

    $forme1 = $anim->add($dessinForme);
    $forme1->moveTo(30, 30);

    $forme2 = $anim->add($dessinForme);
    $forme2->moveTo(20, 60);

    $anim->nextFrame();

    $anim->remove($forme2);
    $anim->nextFrame();

    header("Content-type: application/x-shockwave-flash");
    $anim->output();
?>
```


Tracé et remplissage des formes

Les objets *SWFShape* possèdent des méthodes permettant de fixer les paramètres de leur tracé (épaisseur du trait, couleur) ainsi que les paramètres de remplissage.

SWFShape->setLine()

Précise la largeur et la couleur du tracé de la forme.

Syntaxe	`void setLine(int $largeur, [int $rouge, int $vert, int $bleu], [int $alpha])`
`$largeur`	Largeur du tracé.
`$rouge`	Composante rouge de la couleur du tracé (peut être omise en même temps que `$vert`, `$bleu` et `$alpha` notamment si `$largeur = 0`). Valeur comprise entre 0 et 255.
`$vert`	Composante verte de la couleur du tracé (peut être omise en même temps que `$rouge`, `$bleu` et `$alpha` notamment si `$largeur = 0`). Valeur comprise entre 0 et 255.
`$bleu`	Composante bleue de la couleur du tracé (peut être omise en même temps que `$rouge`, `$vert` et `$alpha` notamment si `$largeur = 0`).

`$alpha`	Composante Alpha (transparence) de la couleur du tracé. Valeur comprise entre 0 (couleur totalement transparente) et 255 (couleur totalement opaque). Ce paramètre est optionnel.

SWFShape->setLeftFill()

Précise le motif de remplissage de l'intérieur de la forme lorsque celle ci est dessinée dans le sens inverse des aiguilles d'une montre (quand l'intérieur se situe à gauche du tracé).

Syntaxe	`void setLeftFill(SWFFill $objetSWFFill)`
`$objetSWFFill`	Objet *SWFFill* précisant le motif de remplissage (voir ci-après).

SWFShape->setRightFill()

Précise le motif de remplissage de l'intérieur de la forme lorsque celle ci est dessinée dans le sens des aiguilles d'une montre (quand l'intérieur se situe à droite du tracé).

Syntaxe	`void setRightFill(SWFFill $objetSWFFill)`
`$objetSWFFill`	Objet *SWFFill* précisant le motif de remplissage (voir ci-après).

Différence entre setLeftFill() et setRightFill()
Nous ne sommes pas parvenus à mettre en évidence la différence de comportement entre `setLeftFill()` et `setRightFill()`. Sur les exemples testés, l'un et l'autre peuvent s'appliquer indifféremment.

Comme vous pouvez le constater, les méthodes de remplissage s'appuient sur des objets que nous n'avons pas encore vus jusque-là. Les objets *SWFFill* ne sont pas censés être instanciés directement. Ils sont en fait retournés par la méthode `addFill()` de l'objet *SWFShape*. Cette méthode accepte trois interfaces distinctes.

SWFShape->addFill() remplissage "solide"

Dans la syntaxe décrite ici, associe à la forme une méthode de remplissage basée tout simplement sur une couleur et retourne l'objet *SWFFill* correspondant.

Syntaxe	`SWFFill addFill(int $rouge, int $vert, int $bleu, [int $alpha])`
`$rouge`	Composante rouge de remplissage de la forme. Valeur comprise entre 0 et 255.

`$vert`	Composante verte de remplissage de la forme. Valeur comprise entre 0 et 255.
`$bleu`	Composante bleue de remplissage de la forme. Valeur comprise entre 0 et 255.
`$alpha`	Composante Alpha (transparence) de la couleur du tracé. Valeur comprise entre 0 (couleur totalement transparente) et 255 (couleur totalement opaque). Ce paramètre est optionnel.

SWFShape->addFill() remplissage par une image

Associe à la forme une méthode de remplissage basée sur un objet "image" (*SWFBitmap*).

Syntaxe	`SWFFill addFill(SWFBitmap $objetSWFBitmap, [int $mode])`
`$objetSWFBitmap`	Objet "image" utilisé comme motif de remplissage (voir ci-après).
`$mode`	Mode de remplissage. Cet argument optionnel peut prendre l'une des deux valeurs définies par des constantes. `SWFFILL_TILED_BITMAP` (valeur par défaut) si vous souhaitez que le motif se répète "à l'infini". `SWFFILL_CLIPPED_BITMAP` si vous souhaitez que le motif n'apparaisse qu'une seule fois. Dans ce cas, avant d'éventuelles transformations par appel aux méthodes proposées par SWFFill, le coin supérieur gauche de l'image coïncide avec l'origine (0,0).

SWFshape->addFill() remplissage par un dégradé

Associe à la forme une méthode de remplissage basée sur un objet "dégradé" (*SWFGradient*).

Syntaxe	`SWFFill addFill(SWFGradient $objetSWFGradient, [int $mode])`
`$objetSWFGradient`	Objet "dégradé" utilisé comme motif de remplissage (voir ci-après).
`$mode`	Mode de remplissage. Cet argument optionnel peut prendre l'une des deux valeurs définies par des constantes. `SWFFILL_LINEAR_GRADIENT` (valeur par défaut) si vous souhaitez que le dégradé soit linéaire (selon l'axe des abscisses x, avant d'éventuelles transformations par appel aux méthodes proposées par *SWFFill*). `SWFFILL_RADIAL_GRADIENT` si vous souhaitez que le dégradé soit radial (par cercles concentriques centrés sur l'origine [0,0], avant d'éventuelles transformations par appel aux méthodes proposées pas *SWFFill*).

Listing 13.32 : ming_fill01.php

```
<?php

    // Création d'une forme quelconque fermée.

    $dessinForme = new SWFShape();

    // Pour mieux apprécié les limites, nous traçons le
    // contour en noir.

    $dessinForme->setLine(1, 0, 0, 0);

    // Et nous remplissons la forme avec la couleur rouge

    $fill = $dessinForme->addFill(255, 0, 0);

    $dessinForme->setLeftFill($fill);

    $dessinForme->movePenTo(-50, -40);
    $dessinForme->drawCurveTo(0, -60, 60, -20);
    $dessinForme->drawCurveTo(10, 5, 60, 40);
    $dessinForme->drawCurveTo(0, 60, -30, 30);
    $dessinForme->drawCurveTo(-20, 0, -50, -40);

    $anim = new SWFMovie();
    $anim->setDimension(300,100);
    $anim->setBackground(255, 255, 255);
    $anim->setRate(1);

    $formePleine=$anim->add($dessinForme);
    $formePleine->moveTo(60, 50);
    $anim->nextFrame();

    header("Content-type: application/x-shockwave-flash");
    $anim->output();
?>
```


Nous voilà maintenant avec deux nouveaux objets restés jusque-là inconnus. Il s'agit des objets *SWFBitmap* et *SWFGradient*.

Le premier s'instancie par un appel au constructeur :

SWFBitmap

Objet image utilisé comme motif de remplissage.

Syntaxe `SWFBitmap SWFBitmap(string $fichier, [string $fichierMasqueAlpha])`

`$fichier`	Nom du fichier image à charger. Ce fichier doit être au format JPEG non progressif ou DBL (Define Dit Lossless). Les fichiers DBL peuvent être créés à partir de fichiers GIF ou PNG grâce aux utilitaires *gif2dbl* et *png2dbl* fournis avec la bibliothèque `Ming`.
`$fichierMasqueAlpha`	Ce paramètre optionnel indique un nom de fichier *.msk*. Ce fichier servira de composante Alpha (transparence) pour l'image JPEG. Les fichiers *.msk* peuvent être créés à partir de fichiers GIF grâce à l'utilitaire *gif2msk* fourni avec la bibliothèque `Ming`.


ATTENTION

Régression ?
Depuis la version 4.2.0 et jusqu'à 4.2.2 (minimum), l'objet SWFBitmap ne semble plus reconnaître aucun format d'image.


REMARQUE

Génération et utilisation des commandes gif2dbl, png2dbl et gif2msk
Les commandes `gif2dbl`, `png2dbl` et `gif2msk` ne sont pas compilées en même temps que la bibliothèque `Ming`. Il faut donc, sous Linux, aller dans le sous répertoire "util" et taper les instructions :

```
# make gif2dbl
# make png2dbl
# make gif2msk
```

pour les générer toutes les trois. De plus, ces fichiers sources nécessitent quelques bibliothèques. (Par exemple, les commandes `gif2` nécessitent le fichier gif_lib.h que vous pourrez trouver dans le `rpm libungif-devel`. Pour cela, consultez votre distribution Linux).
L'utilisation de ces commandes est très simple (elles acceptent toutes le paramètre "--help"):
Pour créer un fichier .dbl dans le même répertoire que le fichier .gif, il suffit de taper :

```
> gif2dbl fichier.gif
```

Pour créer un fichier .dbl dans le même répertoire que le fichier .png, il suffit de taper :

```
> png2dbl fichier.png
```

Pour créer un fichier .msk dans le même répertoire que le fichier .gif, il suffit de taper :

```
> gif2msk fichier.gif
```

Les méthodes de cet objet :

SWFBitmap->getWidth()

Retourne la largeur, en pixels, de l'image.

Syntaxe `int getWidth(void)`

SWFBitmap->getHeight()

Retourne la hauteur, en pixels, de l'image.

Syntaxe `int getWidth(void)`

Voici un exemple de script de remplissage d'une forme par une image répétée à l'infini.

Listing 13.33 : ming_fill02.php

```
<?php

    // Création d'une forme quelconque
    // fermée.

    $dessinForme = new SWFShape();

    // Pour mieux apprécier le tracé
    // nous traçons le contour en noir

    $dessinForme->setLine(1, 0, 0, 0);

    // et nous souhaitons également
    // avoir une forme pleine
    // basée sur un remplissage par
    // une image

    // Donc, dans un premier temps
    // nous créons un objet SWFBitmap
    // en précisant un fichier image
    $bitmap = new SWFBitmap("images/remplissage.jpg");

    // Objet bitmap que nous pouvons alors
    // "déclarér" comme forme de remplissage
    // pour notre forme.
    // Par défaut l'image sera répétée à l'infini
    // Ce qui permet de récupérer un objet
    // SWFFill...
    $fill = $dessinForme->addFill($bitmap);

    // ... qui va nous servir a remplir
    // la forme.
    $dessinForme->setLeftFill($fill);

    // Maintenant que l'on a précisé
    // comme sera rempli la forme
    // on peut la tracer.
    $dessinForme->movePenTo(-50, -40);
    $dessinForme->drawCurveTo(0, -60, 60, -20);
    $dessinForme->drawCurveTo(10, 5, 60, 40);
    $dessinForme->drawCurveTo(0, 60, -30, 30);
```

```
    $dessinForme->drawCurveTo(-20, 0, -50, -40);

    $anim = new SWFMovie();
    $anim->setDimension(300, 100);
    $anim->setBackground(255, 255, 255);
    $anim->setRate(1);

    // Il ne s'agit pas a proprement parler
    // d'une animation puisqu'il n'y qu'une seule
    // vue mais peu importe.
    $formePleine=$anim->add($dessinForme);

    // La forme ayant été dessinée autour
    // de l'origine (0,0) il faut la
    // re-centrer dans l'image
    $formePleine->moveTo(60, 50);

    $anim->nextFrame();

    header("Content-type: application/x-shockwave-flash");
    $anim->output();
?>
```

Ce qui donne le résultat suivant :

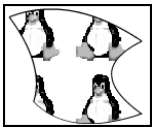


Figure 13.40 :
Remplissage par une image répétée à l'infini

Un autre exemple, où cette fois l'image n'est pas répétée à l'infini.

Listing 13.34 : ming_fill03.php

```
<?php

    // Cet exemple reprend l'exemple précédent.
    // A la seule différence que le mode de
    // remplissage est fixé à SWFFILL_CLIPPED_BITMAP
    // L'image n'est donc pas reproduite à l'infini

    $dessinForme = new SWFShape();

    $dessinForme->setLine(1, 0, 0, 0);

    $bitmap = new SWFBitmap("images/remplissage.jpg");

    // Voilà, c'est juste là où ca diffère.
    $fill = $dessinForme->addFill($bitmap, SWFFILL_CLIPPED_BITMAP);

    $dessinForme->setLeftFill($fill);
```

```
    $dessinForme->movePenTo(-50, -40);
    $dessinForme->drawCurveTo(0, -60, 60, -20);
    $dessinForme->drawCurveTo(10, 5, 60, 40);
    $dessinForme->drawCurveTo(0, 60, -30, 30);
    $dessinForme->drawCurveTo(-20, 0, -50, -40);

    $anim = new SWFMovie();
    $anim->setDimension(300, 100);
    $anim->setBackground(255, 255, 255);
    $anim->setRate(1);

    $formePleine=$anim->add($dessinForme);
    $formePleine->moveTo(60, 50);
    $anim->nextFrame();

    header("Content-type: application/x-shockwave-flash");
    $anim->output();
?>
```

Ce qui donne le résultat suivant (le coin supérieur gauche de l'image coïncide avec l'origine [0,0] de l'objet) :

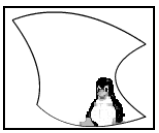


Figure 13.41 :
Remplissage par une image non répétée

Quant au second, il s'instancie par un appel au constructeur :

SWFGradient

Objet "dégradé" utilisé comme motif de remplissage.

Syntaxe `SWFGradient SWFGradient(void)`

et propose l'unique méthode :

SWFGradient->addEntry()

Permet de fixer les couleurs d'un dégradé.

Cette fonction doit être appelée autant de fois que nécessaire pour ajouter les couleurs voulues. Les appels successifs doivent se faire dans l'ordre croissant de `$coef`.

Syntaxe	`void addEntry(double $coef, int $rouge, int $vert, int $bleu, [int $alpha])`
`$coef`	Indique la position de la couleur dans le dégradé. Pour un dégradé linéaire, une valeur `0` indique que le dégradé doit commencer tout à gauche par cette couleur, une valeur de `1` indique que le dégradé doit se terminer tout à droite par cette couleur. Une valeur intermédiaire de `0.3` (par exemple) indique que le dégradé doit passer par cette couleur à son premier tiers (celui du côté gauche). Et évidemment `0.5` indique le milieu. Pour un dégradé radial, une valeur `0` indique que le centre du dégradé doit avoir cette couleur, une valeur de `1` indique que le cercle de plus grand rayon doit avoir cette couleur. Et enfin, toutes les valeurs intermédiaires indiquent des positions intermédiaires pour la couleur. Les dégradés s'étalent approximativement entre les coordonnées x [-800, 800] (pour les dégradés linéaires) ou les rayons [0, 800] (pour les dégradés radiaux).
`$rouge`	Précise la composante rouge de la couleur. Valeur comprise entre 0 et 255.
`$vert`	Précise la composante verte de la couleur. Valeur comprise entre 0 et 255.
`$bleu`	Précise la composante bleue de la couleur. Valeur comprise entre 0 et 255.
`$alpha`	Paramètre optionnel, précise la composante Alpha (transparence) de la couleur. Valeur comprise entre 0 (totalement transparent) et 255 (totalement opaque).

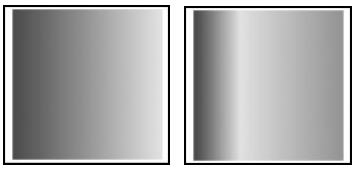


Figure 13.42 : *Exemples de dégradés linéaires (avec deux puis trois couleurs)*

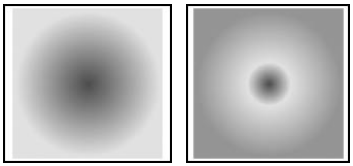


Figure 13.43 : *Exemples de dégradés radiaux (avec deux puis trois couleurs)*

Listing 13.35 : ming_gradient02.php

```
<?php

    // Création d'un carré tout simple
    // afin de le remplir d'un dégradé
    $dessinForme = new SWFShape();

    // Création du dégradé
```

```
// composé de 3 couleurs
$degrade = new SWFGradient();
$degrade->addEntry(0, 255, 0, 0);   // Du Rouge
$degrade->addEntry(0.3, 255, 255, 0); // En pasant par le Jaune
$degrade->addEntry(1, 0, 255, 0);   // Vers le Vert

// On prendra un dégradé linéaire
$fill = $dessinForme->addFill($degrade, SWFFILL_LINEAR_GRADIENT);

$dessinForme->setRightFill($fill);

// Un simple carre de 1700 de coté
// puisque par défaut le dégradé
// s'étale de -800 à 800 (environ)
// Rassurez vous, on verra que l'on
// peut très bien s'affranchir de ce "problème"
$dessinForme->movePenTo(-850, -850);
$dessinForme->drawLine(1700, 0);
$dessinForme->drawLine(0, 1700);
$dessinForme->drawLine(-1700, 0);
$dessinForme->drawLine(0, -1700);

$anim = new SWFMovie();
$anim->setDimension(1750, 1750);
$anim->setBackground(255, 255, 255);
$anim->setRate(1);

$formePleine=$anim->add($dessinForme);
$formePleine->moveTo(875, 875);
$anim->nextFrame();

header("Content-type: application/x-shockwave-flash");
$anim->output();
?>
```



REMARQUE

Méthodes "raccourcis"

Les méthodes `setLeftFill()` *et* `setRightFill()` *possèdent des variantes permettant des raccourcis.*
Ainsi :

- `SWFShape->setLeftFill($rouge, $vert, $bleu, [$alpha])` *est un raccourci pour* `SWFShape->setLeftFill(SWFShape->addFill($rouge, $vert, $bleu, [$alpha]))`.
- `SWFShape->setRightFill($rouge, $vert, $bleu, [$alpha])` *est un raccourci pour* `SWFShape->setRightFill(SWFShape->addFill($rouge, $vert, $bleu, [$alpha]))`.

Afin de mieux maîtriser le positionnement de l'objet de remplissage dans la forme, l'objet *SWFFill* propose un certain nombre de méthodes.

SWFFill->moveTo()

Déplace l'origine du motif de remplissage (par exemple, le coin supérieur gauche de l'image).

Syntaxe	`void moveTo(int $x, int $y)`
`$x`	Nouvelle coordonnée x de l'objet de remplissage.
`$y`	Nouvelle coordonnée y de l'objet de remplissage.

SWFFill->scaleTo()

Modifie la taille du motif de remplissage.

Syntaxe	`void scaleTo(double $coefX, double $coefY)`
`$coefX`	Multiplie la largeur du motif de remplissage par le coefficient `$coefX` par rapport à sa taille initiale. Dans le cas où le motif de remplissage est une image (et non un dégradé), son comportement est plus mystérieux.
`$coefY`	Multiplie la hauteur du motif de remplissage par le coefficient `$coefY` par rapport à sa taille initiale. Dans le cas où le motif de remplissage est une image (et non un dégradé), son comportement est plus mystérieux.

SWFFill->rotateTo()

Fait pivoter le motif de remplissage.

Syntaxe	`void rotateTo(double $angle)`
`$angle`	Angle de rotation du motif de remplissage par rapport à son angle initial, exprimé en degrés.

SWFFill->skewXTo()

Déforme le motif de remplissage selon l'axe des x.

Syntaxe	`void skewXTo(double $pente)`
`$pente`	Coefficient de pente (ou tangente de l'angle d'inclinaison) appliqué sur l'axe des x par rapport à son axe d'origine.

SWFFill->skewYTo()

Déforme le motif de remplissage selon l'axe des y.

Syntaxe	`void skewYTo(double $pente)`
`$pente`	Coefficient de pente (ou tangente de l'angle d'inclinaison) appliqué sur l'axe des y par rapport à son axe d'origine.

Pour mieux positionner une image (non répétée à l'infini), nous avons tout intérêt à faire appel à ces méthodes, comme le montre le script suivant :

Listing 13.36 : ming06.php

```
<?php

    // Cet exemple reprend l'exemple précédent.
    // A la seule différence que l'objet de remplissage
    // est déplacé pour être mieux centré dans la forme

    $dessinForme = new SWFShape();

    $dessinForme->setLine(1, 0, 0, 0);

    $bitmap = new SWFBitmap("images/remplissage.jpg");

    $fill = $dessinForme->addFill($bitmap, SWFFILL_CLIPPED_BITMAP);

    // Jusqu'alors nous n'avons pas manipulé
    // l'objet SWFFill obtenu
    // Et bien, c'est maintenant chose faite
    // Plutot que de faire coincider le coin supérieur
    // gauche de l'image avec l'origine de la forme (0,0)
    // C'est le milieu de l'image que l'on va faire
    // coincider avec l'origine de la forme
    // Ce qui permet également de manipuler les
    // méthodes getWidth() et getHeight() de SWFBitmap
    $fill->moveTo(-$bitmap->getWidth()/2,-$bitmap->getHeight()/2);

    $dessinForme->setLeftFill($fill);

    $dessinForme->movePenTo(-50, -40);
    $dessinForme->drawCurveTo(0, -60, 60, -20);
    $dessinForme->drawCurveTo(10, 5, 60, 40);
    $dessinForme->drawCurveTo(0, 60, -30, 30);
    $dessinForme->drawCurveTo(-20, 0, -50, -40);

    $anim = new SWFMovie();
    $anim->setDimension(300, 100);
    $anim->setBackground(255, 255, 255);
    $anim->setRate(1);
```

```
    $formePleine=$anim->add($dessinForme);
    $formePleine->moveTo(60, 50);
    $anim->nextFrame();

    header("Content-type: application/x-shockwave-flash");
    $anim->output();
?>
```

Ce qui donne le résultat :

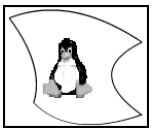


Figure 13.44 : *Image de remplissage centrée*

Ces méthodes sont également indispensables pour bien maîtriser le rendu d'un dégradé.

Listing 13.37 : ming_gradient05.php

```
<?php

    // Création d'une forme quelconque
    // afin de la remplir d'un dégradé
    $dessinForme = new SWFShape();

    // Création du dégradé
    // composé de 2 couleurs
    $degrade = new SWFGradient();
    $degrade->addEntry(0, 255, 0, 0);   // Du Rouge
    $degrade->addEntry(1, 255, 255, 0); // Vers le Jaune

    // On prendra un dégradé linéaire
    $fill = $dessinForme->addFill($degrade, SWFFILL_LINEAR_GRADIENT);

    // Par défaut, le dégradé
    // s'étale des abscisses -800 à 800
    // Notre forme, elle, fait a peut près 60x60
    // Nous allons donc réduire la taille
    // de la forme de remplissage
    $fill->scaleTo(60/1600);

    // Et juste pour le "fun"
    // on va en faire un dégradé à 45 degrés
    $fill->rotateTo(45);

    $dessinForme->setRightFill($fill);

    // Notre forme quelconque
    // de taille approximative
    // 60x60
```

```
    $dessinForme->movePenTo(-50, -40);
    $dessinForme->drawCurveTo(0, -60, 60, -20);
    $dessinForme->drawCurveTo(10, 5, 60, 40);
    $dessinForme->drawCurveTo(0, 60, -30, 30);
    $dessinForme->drawCurveTo(-20, 0, -50, -40);

    $anim = new SWFMovie();
    $anim->setDimension(100, 100);
    $anim->setBackground(255, 255, 255);
    $anim->setRate(1);

    $formePleine=$anim->add($dessinForme);
    $formePleine->moveTo(50, 50);
    $anim->nextFrame();

    header("Content-type: application/x-shockwave-flash");
    $anim->output();
?>
```

Ce qui permet d'obtenir le résultat suivant :

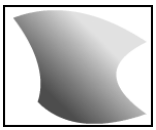


Figure 13.45 : *Remplissage par un dégradé ajusté à la forme*

Le morphing

Il y a un type d'objet dont nous n'avons pas encore parlé jusque-là. Il s'agit de l'objet "morphing". Comme vous l'aurez compris, l'objet morphing permet de réaliser des morphings : autrement dit, cela permet d'afficher une succession de dessins représentant les étapes intermédiaires d'une transition d'un dessin de départ à un dessin d'arrivée par modifications légères. Cet objet s'instancie par un appel au constructeur `SWFMorph()`.

SWFMorph

Objet morphing.

Syntaxe `SWFMorph SWFMorph(void)`

Cet objet propose deux méthodes :

SWFMorph->getShape1()

Retourne une référence sur la forme de départ du morphing afin de pouvoir la dessiner.

Syntaxe `SWFShape getShape1(void)`

SWFMorph->getShape2()

Retourne une référence sur la forme d'arrivée du morphing afin de pouvoir la dessiner.

Syntaxe `SWFShape getShape2(void)`

Le principe d'utilisation de cet objet est donc de récupérer une référence sur l'objet de départ, puis d'utiliser cette référence pour créer l'objet de départ (par appel aux méthodes proposées par l'objet *SWFShape*). Et faire ensuite de même avec l'objet d'arrivée.

L'objet *SWFDisplayItem* propose une méthode dédiée aux objets *SWFMorph*. Il s'agit de la méthode :

SWFDisplayItem->setRatio()

Détermine (pour une vue donnée) quelle étape de la transformation doit être affichée.

Syntaxe `void setRatio(double $etape)`

`$etape` Étape de transformation souhaitée. Cette valeur doit être comprise entre `0` et `1`. Si elle est fixée à `0`, alors c'est l'objet de départ qui est affiché. Si elle est fixée à `1`, alors c'est l'objet d'arrivée qui est affiché. Pour toutes les valeurs intermédiaires, c'est un objet de transition qui est affiché. Cet objet aura d'autant plus l'apparence de l'objet de départ que `$etape` est proche de `0`, et il aura d'autant plus l'apparence de l'objet d'arrivée que `$etape` est proche de `1`.

L'algorithme de base d'une animation de type morphing aura donc l'allure suivante :

Listing 13.38 : ming_morph01.php

```
<?php

    // Instanciation d'un objet
    // de morphing

    $traceMorph = new SWFMorph();

    // Récupération d'une référence
    // sur l'objet de départ
    // du morphing

    $formeDepart = $traceMorph->getShape1();

    // Tracé de la forme de départ
    // par appel aux méthodes
    // de l'objet SWFShape
    // ex: $formeDepart->drawLine(...)
```

```
    // Récupération d'une référence
    // sur l'objet d'arrivée
    // du morphing

    $formeArrivee = $traceMorph->getShape2();

    // Tracé de la forme d'arrivée
    // par appel aux méthodes
    // de l'objet SWFShape
    // ex: $formeArrivee->drawLine(...)

    // Instanciation de l'animation
    // et définition de ses paramètres

    $anim = new SWFMovie();

    // Ajout de l'objet de morphing
    // et récupération de l'objet
    // SWFDisplayItem associé

    $morph=$anim->add($traceMorph);

    // Création des vues successives
    // contenant les différentes
    // phases du morphing
    // En faisant varier le paramètre
    // de setRatio entre 0 et 1

    $nbEtape=30;
    for ($i=0; $i<=$nbEtape; $i++) {
        $morph->setRatio($i/$nbEtape);
        $anim->nextFrame();
    }

    // Emission de l'animation
    // vers le client

    header("Content-type: application/x-shockwave-flash");
    $anim->output();
?>
```

Il est fortement conseillé d'avoir une forme d'arrivée qui contienne le même nombre de segments que la forme de départ. Si la différence est trop grande, vous risquez tout bonnement un "crash" du navigateur. Cependant, même en respectant cette règle, le résultat n'est pas parfait, puisque la forme obtenue avec `setRatio(1)` n'est pas exactement la forme d'arrivée désirée.

Comme c'est le cas, dans l'exemple suivant :

Listing 13.39 : ming_morph02.php

```
<?php

    // Création d'un objet Morphing

    $morph = new SWFMorph();

    // Récuperation d'une référence
    // sur l'objet de départ

    $formeDepart = $morph->getShape1();
    $formeDepart->setLine(1, 0, 0, 0);

    // Dessin de la forme de départ

    // ? inverse (16 segments)
    $formeDepart->movePenTo(-80, -30);
    $formeDepart->drawLine(10, 0);
    $formeDepart->drawLine(0, 10);
    $formeDepart->drawLine(-10, 0);
    $formeDepart->drawLine(0, -10);
    $formeDepart->movePenTo(-80, -10);
    $formeDepart->drawLine(10, 0);
    $formeDepart->drawLine(0, 10);
    $formeDepart->drawLine(-10, 10);
    $formeDepart->drawLine(0, 10);
    $formeDepart->drawLine(10, 0);
    $formeDepart->drawLine(0, -10);
    $formeDepart->drawLine(10, 0);
    $formeDepart->drawLine(0, 20);
    $formeDepart->drawLine(-30, 0);
    $formeDepart->drawLine(0, -20);
    $formeDepart->drawLine(10, -10);
    $formeDepart->drawLine(0, -10);

    // P (10 segments)
    $formeDepart->movePenTo(-50, -30);
    $formeDepart->drawLine(30, 0);
    $formeDepart->drawLine(0, 40);
    $formeDepart->drawLine(-20, 0);
    $formeDepart->drawLine(0, 20);
    $formeDepart->drawLine(-10, 0);
    $formeDepart->drawLine(0, -60);
    $formeDepart->movePenTo(-40, -20);
    $formeDepart->drawLine(10, 0);
    $formeDepart->drawLine(0, 20);
    $formeDepart->drawLine(-10, 0);
    $formeDepart->drawLine(0, -20);

    // H (12 segments)
    $formeDepart->movePenTo(-10, -30);
```

```
$formeDepart->drawLine(10, 0);
$formeDepart->drawLine(0, 20);
$formeDepart->drawLine(10, 0);
$formeDepart->drawLine(0, -20);
$formeDepart->drawLine(10, 0);
$formeDepart->drawLine(0, 60);
$formeDepart->drawLine(-10, 0);
$formeDepart->drawLine(0, -20);
$formeDepart->drawLine(-10, 0);
$formeDepart->drawLine(0, 20);
$formeDepart->drawLine(-10, 0);
$formeDepart->drawLine(0, -60);

// P (10 segments)
$formeDepart->movePenTo(30, -30);
$formeDepart->drawLine(30, 0);
$formeDepart->drawLine(0, 40);
$formeDepart->drawLine(-20, 0);
$formeDepart->drawLine(0, 20);
$formeDepart->drawLine(-10, 0);
$formeDepart->drawLine(0, -60);
$formeDepart->movePenTo(-40, -20);
$formeDepart->drawLine(10, 0);
$formeDepart->drawLine(0, 20);
$formeDepart->drawLine(-10, 0);
$formeDepart->drawLine(0, -20);

// ? (16 segments)
$formeDepart->movePenTo(80, 20);
$formeDepart->drawLine(10, 0);
$formeDepart->drawLine(0, 10);
$formeDepart->drawLine(-10, 0);
$formeDepart->drawLine(0, -10);
$formeDepart->movePenTo(90, 10);
$formeDepart->drawLine(-10, 0);
$formeDepart->drawLine(0, -10);
$formeDepart->drawLine(10, -10);
$formeDepart->drawLine(0, -10);
$formeDepart->drawLine(-10, 0);
$formeDepart->drawLine(0, 10);
$formeDepart->drawLine(-10, 0);
$formeDepart->drawLine(0, -20);
$formeDepart->drawLine(30, 0);
$formeDepart->drawLine(0, 20);
$formeDepart->drawLine(-10, 10);
$formeDepart->drawLine(0, 10);

$formeDepart->movePenTo(-50, 40);
$formeDepart->drawLine(100, 0);
$formeDepart->drawLine(0, 10);
$formeDepart->drawLine(-100, 0);
$formeDepart->drawLine(0, -10);
```

```
$formeArrivee = $morph->getShape2();

$formeArrivee->setLine(1, 0, 0, 0);

// F (10 segments)
$formeArrivee->movePenTo(-120, -30);
$formeArrivee->drawLine(30, 0);
$formeArrivee->drawLine(0, 10);
$formeArrivee->drawLine(-20, 0);
$formeArrivee->drawLine(0, 20);
$formeArrivee->drawLine(10, 0);
$formeArrivee->drawLine(0, 10);
$formeArrivee->drawLine(-10, 0);
$formeArrivee->drawLine(0, 20);
$formeArrivee->drawLine(-10, 0);
$formeArrivee->drawLine(0, -60);

// A (12 segments)
$formeArrivee->movePenTo(-80, -30);
$formeArrivee->drawLine(30, 0);
$formeArrivee->drawLine(0, 60);
$formeArrivee->drawLine(-10, 0);
$formeArrivee->drawLine(0, -20);
$formeArrivee->drawLine(-10, 0);
$formeArrivee->drawLine(0, 20);
$formeArrivee->drawLine(-10, 0);
$formeArrivee->drawLine(0, -60);
$formeArrivee->movePenTo(-70, -20);
$formeArrivee->drawLine(10, 0);
$formeArrivee->drawLine(0, 20);
$formeArrivee->drawLine(-10, 0);
$formeArrivee->drawLine(0, -20);

// C (8 segments)
$formeArrivee->movePenTo(-40, -30);
$formeArrivee->drawLine(30, 0);
$formeArrivee->drawLine(0, 10);
$formeArrivee->drawLine(-20, 0);
$formeArrivee->drawLine(0, 40);
$formeArrivee->drawLine(20, 0);
$formeArrivee->drawLine(0, 10);
$formeArrivee->drawLine(-30, 0);
$formeArrivee->drawLine(0, -60);

// I (8 segments)
$formeArrivee->movePenTo(0, -30);
$formeArrivee->drawLine(10, 0);
$formeArrivee->drawLine(0, 10);
$formeArrivee->drawLine(-10, 0);
$formeArrivee->drawLine(0, -10);
$formeArrivee->movePenTo(0, -10);
```

```
$formeArrivee->drawLine(10, 0);
$formeArrivee->drawLine(0, 10);
$formeArrivee->drawLine(0, 10);
$formeArrivee->drawLine(0, 20);
$formeArrivee->drawLine(-10, 0);
$formeArrivee->drawLine(0, -20);
$formeArrivee->drawLine(0, -10);
$formeArrivee->drawLine(0, -10);

// L (6 segments)
$formeArrivee->movePenTo(30, -30);
$formeArrivee->drawLine(10, 0);
$formeArrivee->drawLine(0, 50);
$formeArrivee->drawLine(20, 0);
$formeArrivee->drawLine(0, 10);
$formeArrivee->drawLine(-30, 0);
$formeArrivee->drawLine(0, -60);

// E (12 segments)
$formeArrivee->movePenTo(70, -30);
$formeArrivee->drawLine(30, 0);
$formeArrivee->drawLine(0, 10);
$formeArrivee->drawLine(-20, 0);
$formeArrivee->drawLine(0, 10);
$formeArrivee->drawLine(10, 0);
$formeArrivee->drawLine(0, 10);
$formeArrivee->drawLine(-10, 0);
$formeArrivee->drawLine(0, 20);
$formeArrivee->drawLine(20, 0);
$formeArrivee->drawLine(0, 10);
$formeArrivee->drawLine(-30, 0);
$formeArrivee->drawLine(0, -60);

// ! (8 segments)
$formeArrivee->movePenTo(110, -30);
$formeArrivee->drawLine(10, 0);
$formeArrivee->drawLine(0, 40);
$formeArrivee->drawLine(-10, 0);
$formeArrivee->drawLine(0, -40);
$formeArrivee->movePenTo(110, 20);
$formeArrivee->drawLine(10, 0);
$formeArrivee->drawLine(0, 10);
$formeArrivee->drawLine(-10, 0);
$formeArrivee->drawLine(0, -10);

$anim = new SWFMovie();
$anim->setDimension(400, 100);
$anim->nextFrame();

$vueMorph=$anim->add($morph);
$vueMorph->moveTo(200, 50);
```

```
    for ($i=0; $i<=20; $i++) {
        $vueMorph->setRatio($i/20);
        $anim->nextFrame();
    }

    header("Content-type: application/x-shockwave-flash");
    $anim->output();
?>
```

Des animations interactives

Il est possible de rendre interactives ces animations en ajoutant des objets boutons. Les boutons sont des objets qui s'instancient par un appel au constructeur `SWFButton`.

SWFButton

Objet bouton.

Syntaxe `SWFButton SWFButton(void)`

Cet objet propose deux méthodes : l'une pour préciser l'aspect visuel du bouton, l'autre pour préciser les actions à déclencher. `SWFButton` propose également une série de méthodes que l'on qualifiera de raccourcis.

L'aspect visuel peut dépendre de l'état du bouton (enfoncé ou non, survolé par le curseur de la souris ou non). On distingue ainsi quatre états, définis par les constantes :

- `SWFBUTTON_UP` : l'état par défaut, c'est-à-dire bouton non enfoncé.
- `SWFBUTTON_OVER` : bouton non enfoncé, mais survolé par le curseur de la souris.
- `SWFBUTTON_DOWN` : bouton enfoncé.
- `SWFBUTTON_HIT`.

Pour définir l'aspect visuel du bouton, on fera donc appel (autant de fois que nécessaire pour préciser la représentation du bouton dans les différents états) à la méthode :

SWFButton->addShape()

Associe une forme à un ou plusieurs états du bouton.

Syntaxe	`void addShape(SWFShape $forme, int $etat))`
`$forme`	Objet qui doit représenter l'objet lorsqu'il est dans l'état précisé.
`$etat`	Un ou plusieurs des états présentés précédemment. Pour préciser plusieurs états, vous utiliserez l'opérateur logique `OU` (ce qui donne par exemple `SWFBUTTON_UP | SWFBUTTON_OVER`).

Qui a donné naissance aux raccourcis :

- `setUp($forme)` équivalant à `addShape($forme, SWFBUTTON_UP)` ;
- `setOver($forme)` équivalant à `addShape($forme, SWFBUTTON_OVER)` ;
- `setDown($forme)` équivalant à `addShape($forme, SWFBUTTON_DOWN)` ;
- `setHit($forme)` équivalant à `addShape($forme, SWFBUTTON_HIT)`.

Pour définir les actions associées à un bouton, nous ferons appel à la méthode :

SWFButton->addAction()

Associe une action à un bouton.

Syntaxe	`void addAction(SWFAction $action, int $evenement)`
`$action`	Action à déclencher.
`$evenement`	Une des valeurs suivantes (ou combinaison avec un `OU` logique) : `SWFBUTTON_MOUSEOVER` : cet événement est déclenché à l'instant où le curseur de la souris entre dans l'objet (bouton non enfoncé). `SWFBUTTON_MOUSEOUT` : cet événement est déclenché à l'instant où le curseur de la souris sort de l'objet (bouton non enfoncé). `SWFBUTTON_MOUSEDOWN` cet événement est déclenché à l'instant où le bouton de la souris est enfoncé, alors que le curseur se situe au-dessus de l'objet. `SWFBUTTON_MOUSEUP` : cet événement est déclenché à l'instant où le bouton de la souris est relâché, alors que le curseur se situe au-dessus de l'objet. C'est généralement sur cet événement que sont déclenchées les actions. `SWFBUTTON_DRAGOUT` : cet événement est déclenché à l'instant où le curseur de la souris quitte l'objet, alors que le bouton a été enfoncé dans l'objet et est maintenu enfoncé lors du déplacement de l'objet. Autrement dit lors d'un cliquer-glisser hors de l'objet. `SWFBUTTON_DRAGOVER` : cet événement est déclenché à l'instant où le curseur de la souris entre dans l'objet, alors que le bouton a été enfoncé dans l'objet et est maintenu enfoncé lors du déplacement de l'objet. Autrement dit lors d'un cliquer-glisser hors puis dans l'objet. `SWFBUTTON_MOUSEUPOUTSIDE` : cet événement est déclenché à l'instant où le bouton est relâché en dehors de l'objet après avoir été enfoncé dans l'objet et maintenu enfoncé lors du déplacement de l'objet. Autrement dit à l'issue d'un cliquer-glisser hors de l'objet.

La méthode `addAction()` possède un raccourci :

- `setAction($action)` équivalant à `addAction($action, SWFBUTTON_MOUSEUP)`.

L'objet action s'instancie en appelant le constructeur `SWFAction()`.

SWFAction

Instancie un objet Action.

Syntaxe	`SWFAction SWFAction(string $script)`
`$script`	Action script a exécuter.


INTERNET

Ressources Javascript
Vous trouverez une liste de sites consacrés à Flash (et à l'Action script) à l'adresse ***http://www.macromedia.com/fr/support/flash/ts/documents/flash_websites.htm****.*
Le document de référence Action script est disponible en anglais à l'adresse : ***http://www.macromedia.com/support/flash/action_scripts/****.*

Les actions peuvent être appliquées directement à l'objet animation (*SWFMovie*) lui-même. Ainsi, une des premières actions que vous serez sans doute amené à demander sera l'arrêt de l'animation (afin, par exemple, de ne pas dérouler toute l'animation sans avoir au préalable cliqué sur un bouton).

Cela se faisant simplement par :

```
<?php
    // (...) creation de l'animation
    $anim->add(new SWFAction("stop();"));
....// (...) génération/affichage de l'animation
?>
```


REMARQUE

Importance du point-virgule
Il est indispensable de terminer toutes les instructions, y compris la dernière, par un point-virgule.

Nous verrons un exemple plus complet d'utilisation de l'Action Script après avoir vu les objets *SWFSprite*.

Les sous-animations

Afin de pouvoir gérer plus facilement les différents éléments d'une animation, il est possible de créer des sous-animations. Ces sous-animations reprennent pour ainsi dire les propriétés d'une animation. Elles peuvent s'instancier par un appel au constructeur `SWFSprite()`.

SWFSprite

Instancie un objet sous-animation.

Syntaxe	`SWFSprite SWFSprite()`
`retour`	Objet *SWFSprite*.

Et elles possèdent les méthodes suivantes :

SWFSprite->add()

Ajoute un objet à la sous-animation.

Syntaxe	`SWFDisplayItem add(mixed $objet)`
`$objet`	Objet à ajouter à la sous-animation. Ce peut être un objet *SWFText*, *SWFTextField*, *SWFShape*, *SWFMorph*, *SWFButton*, *SWFAction* ou *SWFSprite*.
`retour`	Un objet *SWFDisplayItem* uniquement pour les objets graphiques.

SWFSprite->remove()

Supprime un objet de la sous-animation.

Syntaxe	`void remove(SWFDisplayItem $refObjet)`
`$refObjet`	Référence de l'objet à supprimer tel que retournée par la méthode `add()`.

SWFSprite->nextFrame()

Valide la vue courante et passe à la suivante.

Syntaxe	`void nextFrame(void)`

On notera, cependant, qu'il n'est pas possible d'imposer des vitesses de défilement différentes d'une sous-animation à l'autre.

Ces sous-animations se manipulent exactement comme n'importe quel autre objet. Elles s'ajoutent à l'animation principale par appel à la méthode `add()`, et retournent un objet *SWFDisplayItem* qui permet de les déplacer, agrandir, réduire, déformer, etc.

Rappelons qu'il est possible de leur donner un nom grâce à la méthode `setName()` de *SWFDisplayItem*. Ceci est particulièrement utile lorsqu'il s'agit de faire référence à une sous-animation depuis un script "Action script".

Application : jeu du solitaire

Voici, un exemple d'utilisation de l'Action Script faisant intervenir la notion de cliquer-déposer (drag and drop), les sous-animations (ici les pions) et quelques tests conditionnels.

Listing 13.40 : ming_solitaire.php

```
<?php

    $dessinTrou = new SWFShape();
    $dessinTrou->setRightFill(0, 0, 0);
    $dessinTrou->movePenTo(-5, -5);
    $dessinTrou->drawLine(10, 0);
    $dessinTrou->drawLine(0, 10);
    $dessinTrou->drawLine(-10, 0);
    $dessinTrou->drawLine(0, -10);

    $spriteTrou = new SWFSprite();
    $spriteTrou->add($dessinTrou);
    $spriteTrou->nextFrame();

    // Le Pion sera blanc par défaut
    $dessinPion = new SWFShape();
    $dessinPion->setRightFill(255, 255, 255);
    $dessinPion->movePenTo(-4, -4);
    $dessinPion->drawLine(8, 0);
    $dessinPion->drawLine(0, 8);
    $dessinPion->drawLine(-8, 0);
    $dessinPion->drawLine(0, -8);

    // Le Pion sera jaune lorsque le curseur de la souris
    // sera dessus
    $dessinPionSurvole = new SWFShape();
    $dessinPionSurvole ->setRightFill(255, 255, 125);
    $dessinPionSurvole ->movePenTo(-4, -4);
    $dessinPionSurvole ->drawLine(8, 0);
    $dessinPionSurvole ->drawLine(0, 8);
    $dessinPionSurvole ->drawLine(-8, 0);
    $dessinPionSurvole ->drawLine(0, -8);

    // Le Pion sera rouge lorsque le bouton de la souris
    // sera enfoncé
    $dessinPionEnfonce = new SWFShape();
    $dessinPionEnfonce ->setRightFill(255, 0, 0);
    $dessinPionEnfonce ->movePenTo(-4, -4);
    $dessinPionEnfonce ->drawLine(8, 0);
    $dessinPionEnfonce ->drawLine(0, 8);
    $dessinPionEnfonce ->drawLine(-8, 0);
    $dessinPionEnfonce ->drawLine(0, -8);

    $boutonPion = new SWFButton();
```

```
// Déclaration des différents états du pion (bouton)
$boutonPion->addShape($dessinPion, SWFBUTTON_UP | SWFBUTTON_HIT);
$boutonPion->addShape($dessinPionSurvole, SWFBUTTON_OVER);
$boutonPion->addShape($dessinPionEnfonce, SWFBUTTON_DOWN);

// Déclaration des différentes opérations à réaliser
// selon les événements

// Si le bouton est enfoncé, c'est que l'on est en train
// de déplacer le pion
$boutonPion->addAction(new SWFAction("startDrag(this,0);"),
                       SWFBUTTON_MOUSEDOWN);

// Si le bouton est relaché, c'est que l'on est en train
// déposer le pion
$boutonPion->addAction(
        new SWFAction(
            // Arreter le déplacement du pion
            "stopDrag();".
            "deplacementValide = TRUE;".
            // Contrôler où le pion a été déplacé
            // Il doit s'agir d'un trou
            "if (this._droptarget.substr(1,4) != 'trou') {".
            "    deplacementValide = FALSE;".
            "};".
            // Récupération des coordonnées initiales
            // du pion déplacé
            "chaine  = \"\"+this;".
            "tableau = chaine.split('.');".
            "pion    = tableau[tableau.length-1];".
            "pion    = pion.substr(4);".
            "tableau = pion.split('x');".
            "pionx   = parseInt(tableau[0]);".
            "piony   = parseInt(tableau[1]);".
            "if (deplacementValide) {".
            "   trou    = this._droptarget.substr(5);".
            "   tableau = trou.split('x');".
            "   troux   = parseInt(tableau[0]);".
            "   trouy   = parseInt(tableau[1]);".
            // Le trou doit être 2 case a gauche, droite, haut ou bas
            // du point de départ du pion
            "   if (((Math.abs(pionx-troux) != 2)&&".
            "        (Math.abs(piony-trouy) != 2)) ||".
            "       ((Math.abs(pionx-troux)+Math.abs(piony-trouy))>2)) {".
            "        deplacementValide = FALSE;".
            "   };".
            "};".
            "if (deplacementValide) {".
                // Le pion doit sauter un autre pion
                "pionsautex=(troux+pionx)/2;".
                "pionsautey=(trouy+piony)/2;".
```

```
                "if (_level0['pion'+pionsautex+'x'+pionsautey]._visible".
                " != TRUE) {".
                "      deplacementValide = FALSE;".
                "};".
            "};".
            "if (deplacementValide) {".
                // Si c'est ok. Alors on affiche le nouveau pion
                // et on cache le pion déplacé et le pion sauté
                // Affichage du nouveau pion
                "_level0['pion'+troux+'x'+trouy]._visible=TRUE;".
                // Suppression du pion sauté
              "_level0['pion'+pionsautex+'x'+pionsautey]._visible=FALSE;".
                // Suppression du pion déplacé
                "this._visible=FALSE;".
            "};".
            // Quoiqu'il arrive on remet le pion à sa place
            "this._x=12+pionx*12;".
            "this._y=12+piony*12;"
            ),
            SWFBUTTON_MOUSEUP | SWFBUTTON_MOUSEUPOUTSIDE);

$spritePion = new SWFSprite();
$spritePion->add($boutonPion);
$spritePion->nextFrame();

$anim = new SWFMovie();
$anim->setDimension(150, 150);
$anim->setBackground(125, 125, 255);
$anim->setRate(1);
$anim->add(new SWFAction("stop();"));

$larg=3;

// Positionne les trous en croix
for ($x=0; $x<3*$larg; $x++) {
    for ($y=0; $y<3*$larg; $y++) {
        $tierx = floor($x/$larg);
        $tiery = floor($y/$larg);
        if ( (($tierx+$tiery)%2 == 1)
            ||(($tierx == 1) && ($tiery == 1))) {
             $trou[$x][$y] = $anim->add($spriteTrou);
             $trou[$x][$y]->setName("trou".$x."x".$y);
             $trou[$x][$y]->moveTo(12+$x*12,12+$y*12);
        }
    }
}

// Positionne tous les pions
// à tous les endroits possibles
for ($x=0; $x<3*$larg; $x++) {
    for ($y=0; $y<3*$larg; $y++) {
        $tierx = floor($x/$larg);
```

```
            $tiery = floor($y/$larg);
            if ( (($tierx+$tiery)%2 == 1)
                ||(($tierx == 1) && ($tiery == 1))) {
                 $pion[$x][$y] = $anim->add($spritePion);
                 $pion[$x][$y]->setName("pion".$x."x".$y);
                 $pion[$x][$y]->moveTo(12+$x*12,12+$y*12);
            }
        }
    }

    // Masque le pion du milieu
    $anim->add(new SWFAction("pion4x4._visible=FALSE;"));

    $anim->nextFrame();

    header("Content-type: application/x-shockwave-flash");
    $anim->output();
?>
```

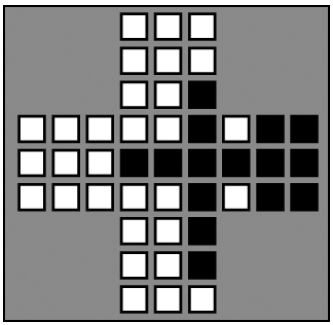


Figure 13.46 : *Un solitaire en Flash*

Chapitre 14

La création de documents PDF

PHP permet de créer des documents PDF à la volée. Il est ainsi possible de créer ce type de documents à partir d'informations récupérées d'un formulaire ou d'une base de données. Pour cela, nous disposons notamment de la bibliothèque `PDFLib`.

`PDFLib` est gratuite sauf dans le cas d'un usage commercial.

Cette bibliothèque est toujours en développement et de nombreuses fonctions ont été modifiées. Le peu de documentation disponible et les changements soudains dans le nom et les paramètres des fonctions peuvent causer des problèmes d'une version à l'autre de `PDFLib`. Mais cela ne devrait pas vous empêcher d'utiliser cette bibliothèque.

14.1. Installation

Sous Windows

Avec l'archive du PHP Group

Avec PHP 5, Vous devez copier le fichier *php_pdf.dll* disponible dans le paquetage PECL du PHP Group dans le répertoire des extensions PHP. Avec PHP 4, pas de soucis, la DLL fait déjà parti de l'archive de PHP Group.

Dans tous les cas, il vous faut ensuite modifier le fichier *php.ini* pour ajouter ou décommenter une ligne :

```
extension=php_pdf.dll
```

Vous pouvez vous reporter au chapitre "Prise en main" pour plus de détails sur l'installation du paquetage PECL.

Avec EasyPHP

Le support de PDF est activé par défaut avec EasyPHP.

Sous Linux

L'extension PDF n'est plus disponible avec les sources de la version 5 de PHP.

Avec PHP4, vous devez, dans un premier temps, vous procurer la bibliothèque `PDFLib` disponible sur le site (en anglais) **http://www.pdflib.com/** (il s'agit du fichier *PDFlib-5.0.1-Linux.tar.gz*).

Après avoir copié l'archive dans un répertoire quelconque (disons */usr/local/src/lib*), vous devez la décompresser.

```
# gunzip PDFlib-5.0.1-Linux.tar.gz
# tar xvf PDFlib-5.0.1-Linux.tar
```

Il vous est ensuite possible de lancer la compilation.

```
# ./configure
```

```
# make
# make install
```

Vous disposez désormais d'une série de fichiers *libpdf** sous le répertoire */usr/local/lib*.

Il vous suffit alors de recompiler PHP avec l'option `--with-pdflib=/usr/local`.

Vous pouvez vous reporter au chapitre "Prise en main" pour plus de détails sur la compilation de PHP.

Les polices d'écriture

Pour utiliser une police d'écriture, celle-ci devra être disponible et déclarée dans un fichier *pdflib.upr*.

Quelques polices d'écriture et un fichier *pdflib.upr* sont fournis avec la bibliothèque (sous le répertoire *fonts*). Vous pouvez copier ce répertoire dans un endroit quelconque (*/usr/local/fonts* par exemple). Pour utiliser les polices fournies, vous devrez modifier le fichier *pfdlib.upr* en décommentant et modifiant le chemin précisé pour les polices. Ce qui, dans notre cas, donnerait (le slash devant le chemin est important) :

```
//usr/local/fonts
```

Si vous disposez d'autres polices d'écriture, vous devrez compléter ce fichier en conséquence.

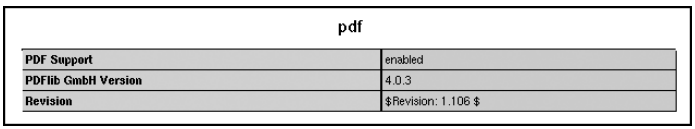
Le chemin vers ce fichier doit être déclaré dans une variable d'environnement `PDFLIBRESOURCE`.

Idéalement, cette variable devra être définie dans le compte sous lequel le serveur tourne. Il est toutefois possible de la déclarer au niveau des scripts PHP.

```
putEnv("PDFLIBRESOURCE=/chemin_vers_pdflib_url/pdflib.url");
```

Vérification

Pour vérifier l'activation du support de `PDFLib`, appelez l'habituel script contenant `<?php phpinfo(); ?>`. Celui-ci doit afficher :



pdf

PDF Support	enabled
PDFlib GmbH Version	4.0.3
Revision	$Revision: 1.106 $

Figure 14.1 : *phpinfo()*

14.2. Créer la base d'un document PDF

La création d'un document PDF passe par un certain nombre d'étapes obligatoires.

Il faut tout d'abord créer un "objet" (il ne s'agit toutefois pas d'un véritable objet, mais plutôt d'une ressource) grâce à la fonction `pdf_new()`. Les ressources ainsi allouées devront, au final, être libérées par un appel à la fonction `pdf_delete()`.

pdf_new()

Crée un nouvel objet PDF en lui allouant la mémoire nécessaire.

Syntaxe	`resource pdf_new(void)`
`retour`	Un identifiant d'objet PDF.

pdf_delete()

Libère les ressources allouées pour un objet PDF.

Syntaxe	`boolean pdf_delete(resource $objetPDF)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`retour`	`TRUE`.

Une fois un objet PDF créé, vous devrez créer un document PDF. Pour cela, vous disposez de la fonction `pdf_open_file()` qui vous permettra de créer, au choix, un fichier ou le document en mémoire. Le document ainsi généré devra être validé par un appel à `pdf_open_close()`.

pdf_open_file()

Ouvre un nouveau document PDF (sous forme de fichier ou en mémoire).

Syntaxe	`boolean pdf_open_file(resource $objetPDF , string $nomFichier)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$nomFichier`	Nom et chemin absolu du fichier à créer. Si cet attribut est une chaîne vide, alors le document sera juste créé en mémoire. Avec PHP4, ce paramètre était tout simplement optionnel.
`retour`	`TRUE`.

pdf_close()

Ferme le document PDF.

Syntaxe	`boolean pdf_close(resource $objetPDF)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`retour`	`TRUE`.

Si vous avez opté pour un document créé en mémoire, vous devrez obligatoirement faire appel à la fonction `pdf_get_buffer()` après l'appel à `pdf_close()`, et avant celui à `pdf_delete()`.

pdf_get_buffer()

Permet de récupérer le contenu de la mémoire d'un objet PDF quand celui-ci n'a pas été sauvegardé dans un fichier.

Syntaxe	`string pdf_get_buffer(resource $objetPDF)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`retour`	Le fichier PDF dans une chaîne de caractères.

Et, enfin, chaque document doit nécessairement contenir au moins une page. L'ajout et la création d'une page débute par un appel à `pdf_begin_page()` et se termine par un appel à `pdf_end_page()`.

pdf_begin_page()

Créer une nouvelle page.

Syntaxe	`boolean pdf_begin_page(resource $objetPDF, double $largeur, double $hauteur)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$largeur`	Largeur en points.
`$hauteur`	Hauteur en points.
`retour`	`TRUE`.

Voici un tableau de correspondances en points des formats les plus courants :

Tableau 14.1 : Les tailles en points des formats courants de papier

Format	Taille en points
A0 (84 x 118,8 cm)	2380 x 3368
A1 (59,4 x 84 cm)	1684 x 2380
A2 (42 x 59,4 cm)	1190 x 1684
A3 (29,7 x 42 cm)	842 x 1190
A4 (21 x 29,7 cm)	595 x 842
A5 (14,85 x 21 cm)	421 x 595
A6 (10,5 x 14.85 cm)	297 x 421

Format	Taille en points
B5 (17.67 x 25 cm)	501 x 709
Letter (8.5" x 11")	612 x 792
Legal (8.5" x 14")	612 x 1008
Ledger (17" x 11")	1224 x 792

pdf_end_page()

Termine la page courante.

Syntaxe	`boolean pdf_end_page(resource $objetPDF)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`retour`	`TRUE`.

Le squelette d'un script de génération d'un document PDF aura donc l'allure suivante dans le cas de la génération d'un fichier :

```
<?php
    $pdf = pdf_new();
    pdf_open_file($pdf, "monFichier.pdf");
    // Appel aux fonctions précisant les informations
    // liés au fichier
    // Puis autant de pdf_begin_page() pdf_end_page() que nécessaire
    pdf_begin_page($pdf, 595, 842);   // Page en A4
        // Contenu de la page
    pdf_end_page($pdf);
    pdf_close($pdf);
    pdf_delete($pdf);
?>
```

Et il aura l'allure suivante dans le cas d'un document créé en mémoire et envoyé au navigateur :

```
<?php
    $pdf = pdf_new();
    pdf_open_file($pdf, "");
    // Appel aux fonctions précisant les informations
    // liés au fichier
    // Puis autant de pdf_begin_page() pdf_end_page() que nécessaire
    pdf_begin_page($pdf, 595, 842);   // Page en A4
        // Contenu de la page
    pdf_end_page($pdf);
    pdf_close($pdf);
    $donnees = pdf_get_buffer($pdf);
    pdf_delete($pdf);
```

```
header("Content-type: application/pdf");
header("Content-length: ".strlen($donnees));
header("Content-disposition: inline; filename=test.pdf");
echo $donnees;

?>
```

Il est évidemment possible d'envoyer le document au navigateur, même lorsque celui-ci est stocké dans un fichier. Il suffit, pour cela, de lire le contenu du fichier, comme nous le ferons dans le prochain exemple.

14.3. Préciser les attributs (mots-clés) du document

Pour chaque document PDF, il est possible de définir des mots-clés, le nom de l'auteur, etc.

pdf_set_info()

Permet de remplir les informations d'un fichier PDF.

Syntaxe	`boolean pdf_set_info(resource $objetPDF, string $cle, string $valeur)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$cle`	Pouvant prendre les valeurs prédéfinies `"Subject"`, `"Title"`, `"Creator"`, `"Author"`, `"Keywords"` ou une valeur définie par l'utilisateur (à l'exception des mots réservés `"CreationDate"`, `"Producer"`, `"ModDate"` et `"Trapped"`).
`$value`	Valeur associée à la clé.
`retour`	`TRUE`.

L'appel à `pdf_set_info()` se fera en dehors de la création d'une page (en dehors d'un bloc `pdf_begin_page()`, `pdf_end_page()`).

14.4. Préciser les paramètres de la page

Pour chaque page du document, il est possible de définir (ou redéfinir) les paramètres devant être utilisés.

pdf_set_value()

Permet d'attribuer une valeur aux paramètres de la page (taux de compression, interligne, etc).

Syntaxe	`boolean pdf_set_value(resource $objetPDF, string $parametre, double $valeur)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$parametre`	Paramètre à définir.
`$valeur`	Valeur à attribuer.
`retour`	`TRUE`.

Voici un tableau des valeurs pouvant être modifiées :

Tableau 14.2 : Valeurs pouvant être modifiées par pdf_set_value()

Clé	Valeur	Valeur par défaut
`compress`	Définit la compression du document entre `0` et `9`. `0` indique qu'il n'y a pas de compression tandis que `9` est la compression la plus forte.	`6`
`pagewidth`	Modifie la largeur de la page courante.	
`pagewidth`	Modifie la largeur de la page courante.	
`pageheight`	Modifie la hauteur de la page courante.	
`leading`	Modifie l'espace entre deux lignes de base de deux lignes consécutives.	
`Textrise`	Modifie l'espace entre la position désirée du texte et la ligne de base.	`0` (et remis à `0` à chaque nouvelle page).
`textrendering`	`0` pour du texte plein. `1` pour n'avoir que le contour du texte. `2` pour du texte plein et le contour. `3` pour ne pas afficher le texte, mais l'ajouter au chemin.	`0`
`Horizscaling`	Pourcentage pour élargir ou rétrécir le texte.	`100`
`charspacing`	Espace entre deux caractères à ajouter à l'espacement par défaut.	`0`
`wordspacing`	Espace à ajouter à la taille du caractère d'espacement.	`0`
`duration`	Définit le temps d'affichage en secondes pour la page courante.	`1`

Il est également possible d'ajouter des signets aux pages.

pdf_add_bookmark()

Ajoute un signet à une page.

Syntaxe	`int pdf_add_bookmark(resource $objetPDF, string $texte, int $parent, boolean $ouvrir)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$texte`	Label du signet.
`$parent`	Identifiant du signet parent (`0` revient à créer un signet à la racine de l'arborescence). Ce paramètre est optionnel avec PHP4.
`$ouvrir`	Si cet argument vaut `TRUE`, alors les enfants du signet seront visibles. Ce paramètre est optionnel avec PHP4.
`retour`	Identifiant de signet (pouvant servir de paramètre parent pour un autre signet).

14.5. Afficher du texte

Avant d'afficher du texte, vous devrez sélectionner une police d'écriture. Vous ne pouvez toutefois sélectionner une police d'écriture que si celle-ci a été précédemment chargée (ou tout au moins ajoutée à la liste des polices utilisées).

pdf_findFont()

Ajoute une police d'écriture à la liste des polices.

Syntaxe	`int pdf_findFont(resource $objetPDF, string $nomPolice, string $encodage, boolean $integrer)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$nomPolice`	Nom de la police d'écriture à utiliser.
`$encodage`	"`builtin`", "`macroman`", "`winansi`", "`ebcdic`", "`host`" ou un nom défini par l'utilisateur.
`$integrer`	`TRUE` si la police doit être intégrée au document , `FALSE` sinon (le lecteur devra avoir la police à sa disposition).
`retour`	Un identifiant de police d'écriture, ou `FALSE`.

pdf_setFont()

Définit la police d'écriture courante.

Syntaxe	`boolean pdf_setFont(resource $objetPDF, int $police, double $taille)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$police`	Identifiant tel que retourné par `pdf_findFont()`.
`$taille`	Taille de la police (en points).
`retour`	`TRUE`.

Une fois ce travail effectué, vous pouvez commencer à envisager d'ajouter du texte à votre document grâce, notamment, à la fonction `pdf_show_xy()`.

pdf_show_xy()

Affiche du texte à une position définie par ses coordonnées.

Syntaxe	`boolean pdf_show_xy(resource $objetPDF, string $texte, double $x, double $y)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$texte`	Texte à afficher.
`$x`	Abscisse du bord gauche du texte (l'origine du document est en bas à gauche de la page).
`$y`	Ordonnée de la ligne de base du texte (l'origine du document est en bas à gauche de la page).
`retour`	`TRUE`.

Les fonctions que nous avons vues jusqu'à présent nous permettent de créer le script (assez simple) suivant :

Listing 14.1 : pdf1.php

```
<?php
$nomFichier="test.pdf";
// Creation d'un pdf
$pdf = pdf_new();
// Ouverture d'un fichier en ecriture le nom
// du fichier est un chemin absolu.
pdf_open_file($pdf, $nomFichier);

// Definition de l'auteur du document
pdf_set_info($pdf, "Author", "Thomas, Damien, Laurent et Pem");
```

```
// Definition du titre
pdf_set_info($pdf, "Title", "Mon premier PDF en PHP");
// Definition du createur du PDF
pdf_set_info($pdf, "Creator", "The PHP bible team :)");
// Definition du sujet
pdf_set_info($pdf, "Subject", "PDF en PHP");

// Creation d'une page de 595 points par 842 (A4)
// (72 points par pouce soit 28,368 par cm)
pdf_begin_page($pdf, 595, 842);
// Donne le label "La page" a la page
pdf_add_bookmark($pdf, "La page", 0, TRUE);
// Definit la police courante
$font = pdf_findfont($pdf, "Courier", "host", FALSE);
if ($font) {
    pdf_setfont($pdf, $font, 30);
} else {
    die ("Police d'ecriture introuvable");
}// Definit la valeur de textrendering
pdf_set_value($pdf, "textrendering", 1);
// Affiche le texte "Trop trop cool ce truc" aux coordonnees (50,750)
// (50,750) est le point inferieur gauche de la ligne de base
// La ligne de base est la ligne inferieur naturelle
// (p,q,y... descendent en dessous)
// Le point (0,0) est le point inferieur gauche du document)
pdf_show_xy($pdf, "Trop trop cool ce truc", 50, 750);

// Termine la page
pdf_end_page($pdf);
// Ferme le document PDF
pdf_close($pdf);
// Efface l'objet PDF de la memoire et les ressources associees
pdf_delete($pdf);

/************************************/
/* Cette partie concerne l'affichage */
/* du document, le fichier est deja  */
/* cree sur le disque dur.           */
/************************************/
$taille = filesize($nomFichier);
header("Content-type: application/pdf");
header("Content-Length: $taille");
header("Content-Disposition: inline; filename=test.pdf");
readfile($nomFichier);
?>
```

Dont l'affichage serait :

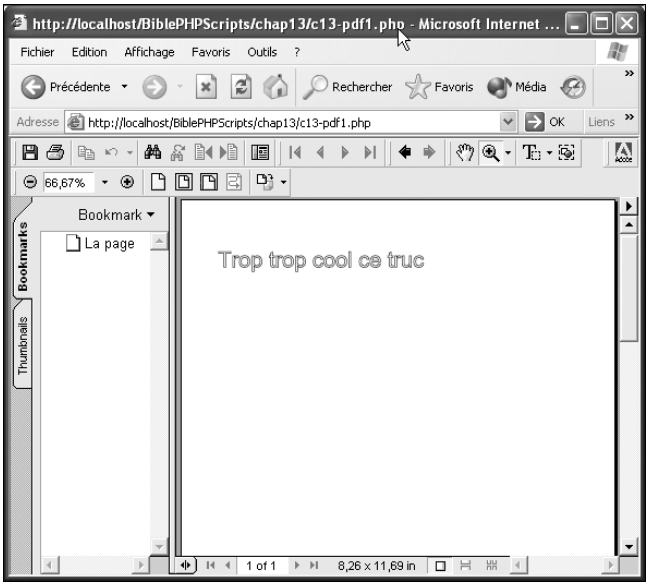


Figure 14.2 : *Écrire du texte dans un fichier PDF*

Il est également possible d'écrire du texte à la position courante du pointeur de texte, ce qui permet, en particulier, d'ajouter du texte à la suite du texte précédent.

pdf_show()

Affiche du texte à la position courante du pointeur.

Syntaxe	`boolean pdf_show(resource $objetPDF, string $texte)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$texte`	Texte à afficher.
`retour`	`TRUE`.

La position courante du pointeur de texte peut être modifiée à tout instant grâce à la fonction `pdf_set_text_pos()`.

pdf_set_text_pos()

Modifie la position courante du pointeur de texte.

Syntaxe	`boolean pdf_set_text_pos(resource $objetPDF, double $x, double $y)`

`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$x`	Abscisse du pointeur.
`$y`	Ordonnée du pointeur.
`retour`	`TRUE`.

Vous n'aurez toutefois pas besoin d'avoir recours à des calculs savants pour écrire du texte sur une nouvelle ligne.

pdf_continue_text()

Affiche du texte à la ligne suivante. L'espace entre deux lignes est déterminé par la variable `leading` qui peut être modifiée par `pdf_set_value()`.

Syntaxe	`void pdf_continue_text(int $objetPDF, string $texte)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$texte`	Texte à afficher.

De même, l'écriture sur plusieurs colonnes se trouve grandement simplifiée par la fonction `pdf_show_boxed()`.

pdf_show_boxed()

Permet d'afficher un texte dans un rectangle virtuel ; l'alignement et les retours à la ligne seront automatiquement gérés pour être ajustés au rectangle.

Syntaxe	`int pdf_show_boxed(resource $objetPDF, string $texte, double $gauche, double $haut, double $largeur, double $hauteur, string $alignement, string $option)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$texte`	Texte à afficher.
`$gauche`	Abscisse du bord gauche.
`$haut`	Ordonnée du bord supérieur.
`$largeur`	Largeur du rectangle (positionner largeur et hauteur à `0` revient à écrire sur une ligne).
`$hauteur`	Hauteur du rectangle (positionner largeur et hauteur à `0` revient à écrire sur une ligne).
`$alignement`	Au choix : `"left"` pour un alignement à gauche. `"right"` pour un alignement à droite. `"center"` pour centrer le texte.

	"justify" pour un texte justifié (sauf la dernière ligne qui sera alignée à gauche). "fulljustify" pour un texte justifié (y compris la dernière ligne).
$option	"blind" si vous ne souhaitez pas afficher le texte mais simplement faire un test (pour, par exemple, déterminer si avec les valeurs choisies tout le texte rentre dans le rectangle). Ce paramètre est optionnel avec PHP 4.
retour	Le nombre de caractères qui n'ont pu être contenus dans le rectangle.

Un petit penchant pour l'écriture inclinée ?
Si vous souhaitez écrire un texte incliné, il vous suffit de faire pivot sur les axes des coordonnées (comme nous le verrons plus loin).

Largeur d'un texte

pdf_stringWidth()

Retourne la largeur d'une chaîne de caractères en points en utilisant, par défaut, la police d'écriture courante.

Syntaxe	`double pdf_stringWidth(resource $objetPDF, string $texte, int $police, double $taille)`
$objetPDF	Identifiant tel que retourné par `pdf_new()`.
$texte	Texte dont on veut connaître la largeur.
$police	Identifiant de police de caractères tel que retourné par `pdf_findFont()`. Ce paramètre est optionnel avec PHP 4.
$taille	Taille de la police d'écriture. Ce paramètre est optionnel avec PHP 4.
retour	La largeur de la chaîne de caractères en utilisant la police d'écriture définie.

Les paramètres de la police d'écriture

pdf_get_parameter()

Retourne des informations liées à la police de caractères utilisée.

Syntaxe	`string pdf_get_parameter(int $objetPDF, string $cle, double $remplace)`

`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$cle`	Nom du paramètre à récupérer.
`$remplace`	Valeur de remplacement. Ce paramètre est optionnel avec PHP 4.
`retour`	La valeur du paramètre.

Tableau 14.3 : Paramètre récupérable par pdf_get_parameter()

Clé	Description
`fontencoding`	Récupère le nom de l'encodage de la police d'écriture prédéfinie par `pdf_set_font()`.
`underline`	Retourne la chaîne de caractères "`true`" si le texte est souligné, "`false`" sinon.
`overline`	Retourne la chaîne de caractères "`true`" si le texte est surligné, "`false`" sinon.
`strikeout`	Retourne la chaîne de caractères "`true`" si le texte est barré, "`false`" sinon.

14.6. Dessiner des formes géométriques

Maintenant que nous avons vu comment écrire du texte dans un document PDF, voyons comment nous pouvons dessiner.

Le tracé se fait en dessinant des cercles, des rectangles, en traçant des segments entre la position courante du curseur et un point quelconque, en modifiant la position courante du curseur, etc. Mais, quoi qu'il arrive, le tracé doit être validé par un appel aux fonctions `pdf_*_stroke()` (pour tracer le contour), `pdf_*_fill()` (pour remplir la forme ainsi tracée) ou `pdf_*_fill_stroke()` (pour remplir et tracer le contour de la forme).

Les lignes

Pour tracer une ligne, vous serez sans doute amené à déplacer le curseur.

pdf_moveTo()

Définit la nouvelle position du pointeur graphique.

Syntaxe	`boolean pdf_moveTo(resource $objetPDF, double $x, double $y)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$x`	Abscisse du pointeur graphique.
`$y`	Ordonnée du pointeur graphique.
`retour`	`TRUE`.

pdf_lineTo()

Trace un segment depuis la position du pointeur graphique jusqu'aux coordonnées passées en paramètre.

Syntaxe	`boolean pdf_lineTo(resource $objetPDF, double $x, double $y)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$x`	Abscisse de la fin du segment.
`$y`	Ordonnée de la fin du segment.
`retour`	`TRUE`.

Les rectangles

pdf_rect()

Permet de dessiner un rectangle (le pointeur graphique prend alors la position du coin inférieur gauche du rectangle).

Syntaxe	`boolean pdf_rect(resource $objetPDF, double $x, double $y, double $largeur, double $hauteur)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$x`	Abscisse du bord gauche du rectangle.
`$y`	Ordonnée du bord inférieur du rectangle.
`$largeur`	Largeur du rectangle.
`$hauteur`	Hauteur du rectangle.
`retour`	`TRUE`.

Les cercles et arcs de cercle

pdf_circle()

Dessine un cercle (le pointeur graphique prend alors la position de l'extrémité droite du cercle).

Syntaxe	`boolean pdf_circle(resource $objetPDF, double $x, double $y, double $rayon)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$x`	Abscisse du centre de l'arc.
`$y`	Ordonnée du centre de l'arc.

`$rayon`	Rayon de l'arc.
`retour`	`TRUE`.

pdf_arc()

Dessine un arc de cercle dans le sens trigonométrique (un segment reliera la position courante du pointeur à la première extrémité de l'arc, et le pointeur graphique sera déplacé à la seconde extrémité de l'arc).

Syntaxe	`boolean pdf_arc(resource $objetPDF, double $x, double $y, double $rayon, double $angle1, double $angle2)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$x`	Abscisse du centre de l'arc.
`$y`	Ordonnée du centre de l'arc.
`$rayon`	Rayon de l'arc.
`$angle1`	Angle de départ (en degrés).
`$angle2`	Angle de fin (en degrés).
`retour`	`TRUE`.

pdf_arcn()

Dessine un arc de cercle dans le sens horaire (un segment reliera la position courante du pointeur à la première extrémité de l'arc, et le pointeur graphique sera déplacé à la seconde extrémité de l'arc).

Syntaxe	`void pdf_arcn(resource $objetPDF, double $x, double $y, double $rayon, double $angle1, double $angle2)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$x`	Abscisse du centre de l'arc.
`$y`	Ordonnée du centre de l'arc.
`$rayon`	Rayon de l'arc.
`$angle1`	Angle de départ (en degrés).
`$angle2`	Angle de fin (en degrés).
`retour`	`TRUE`.

Les courbes de Bézier

pdf_curveTo()

Permet de tracer une courbe de Bézier cubique entre la position courante du pointeur graphique et les coordonnées du dernier point précisé (qui deviendra la nouvelle position du pointeur) en s'appuyant sur deux autres points.

Syntaxe	`boolean pdf_curveto(resource $objetPDF, double $x1, double $y1, double $x2, double $y2, double $x3, double $y3)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$x1`	Abscisse du premier point d'appui
`$y1`	Ordonnée du premier point d'appui.
`$x2`	Abscisse du deuxième point d'appui.
`$y2`	Ordonnée du deuxième point d'appui.
`$x3`	Abscisse du point d'arrivée.
`$y3`	Ordonnée du point d'arrivée.
`retour`	`TRUE`.

Clôture et validation du tracé

Si vous souhaitez fermer la figure que vous venez de dessiner, sans avoir à préciser les coordonnées du point initial du tracé, vous devez faire appel à `pdf_closePath()`.

pdf_closePath()

Permet de clore un tracé.

Syntaxe	`boolean pdf_closePath(resource $objetPDF)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`retour`	`TRUE`.

Toutefois, un appel à `pdf_closePath()` ne valide pas le tracé du contour (opération qui doit nécessairement être faite avant l'appel à `pdf_end_page()`), contrairement aux fonctions qui suivent.

pdf_closepath_stroke()

Permet de clore un tracé et d'afficher le contour.

Syntaxe	`boolean pdf_closepath_stroke(resource $objetPDF)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`retour`	`TRUE`.

pdf_closePath_fill_stroke()

Permet de clore un tracé, de le remplir et d'afficher le contour (les couleurs du remplissage et du contour étant définies par `pdf_selColor()`).

Syntaxe	`boolean pdf_closePath_fill_stroke(resource $objetPDF)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`retour`	`TRUE`.

Si, toutefois, vous considérez que votre tracé est fermé, vous pouvez appeler les fonctions suivantes :

pdf_stroke()

Permet d'afficher le tracé.

Syntaxe	`boolean pdf_stroke(resource $objetPDF)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`retour`	`TRUE`.

pdf_fill()

Permet de remplir un tracé.

Syntaxe	`boolean pdf_fill(resource $objetPDF)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`retour`	`TRUE`.

pdf_fill_stroke()

Permet de remplir un tracé et d'en afficher le contour (les couleurs du remplissage et du contour étant définies par `pdf_selColor()`).

Syntaxe	`boolean pdf_fill_stroke(resource $objetPDF)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`retour`	`TRUE`.

14.7. Modifier les paramètres du tracé

Il est, bien entendu, possible de jouer sur les paramètres du tracé (comme la couleur, la largeur du tracé, etc.). Mais attention, ceci ne peut se faire que si le tracé n'a pas été commencé (i.e. juste après avoir validé le tracé précédent).

La couleur

pdf_setColor()

Définit la couleur de trait et/ou de remplissage.

Syntaxe	`boolean pdf_setColor(resource $objetPDF, string $type, string $palette, numeric $c1, double $c2, double $c3, double $c4)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$type`	Élément dont vous souhaitez changer la couleur : "`fill`" pour changer la couleur de remplissage. "`stroke`" pour changer la couleur du trait. "`both`" pour changer les deux (et leur donner la même couleur).
`$palette`	Précise comment vous souhaitez identifier la couleur : "`gray`" si vous souhaitez indiquer un niveau de gris. "`rgb`" si vous souhaitez préciser les composantes rouge, verte et bleue de la couleur. "`cmyk`" si vous souhaitez préciser les composantes cyan, magenta, jaune et noire de la couleur. "`spot`" si vous souhaitez préciser l'identifiant de la couleur (dans le cas d'une couleur définie par `pdf_makeSpotColor()` décrite ci-après). "`pattern`".
`$c1`	Selon les cas : Niveau de gris (`0` = noir, `1` = blanc) dans le cas "`gray`". Composante rouge (entre `0` et `1`) dans le cas "`rgb`". Composante cyan (entre `0` et `1`) dans le cas "`cmyk`".

	Identifiant de la couleur dans le cas "spot". Indice dans la palette dans le cas "pattern".
$c2	Selon les cas : (Ce paramètre est optionnel avec PHP4) Coefficient à appliquer à la couleur (entre 0 = noir et 1 = couleur sauvegardée) dans le cas "spot". Composante verte (entre 0 et 1) dans le cas "rgb". Composante magenta (entre 0 et 1) dans le cas "cmyk"
$c3	Selon les cas : (Ce paramètre est optionnel avec PHP4) Composante bleue (entre 0 et 1) dans le cas "rgb". Composante jaune (entre 0 et 1) dans le cas "cmyk"
$c4	Composante noire (entre 0 et 1, uniquement dans le cas "cmyk"). Ce paramètre est optionnel avec PHP4.
retour	TRUE.

Quelques exemples :

```
// Définit la couleur de remplissage à bleu
pdf_setColor($pdf, "fill", "rgb", 0, 0, 1);
// Définit la couleur de trait à gris clair
pdf_setColor($pdf, "stroke", "gray", 0.8);
// Définit la couleur de trait et de remplissage à rouge
pdf_setColor($pdf, "both", "cmyk", 1, 0, 0, 0.7);
```

Comme cela a été évoqué au cours de la description de la fonction précédente, il est possible d'attribuer un identifiant à une couleur (ce qui est plus simple d'utilisation). Pour cela, il faut utiliser la fonction pdf_makeSpotColor().

pdf_makeSpotColor()

Retourne un identifiant pour la couleur de remplissage courante.

Syntaxe	int pdf_makespotcolor(resource $objetPDF, string $nom)
$objetPDF	Identifiant tel que retourné par pdf_new().
$nom	Nom à donner à la couleur.
retour	Identifiant de la couleur.

La largeur, les extrémités et les bords

pdf_setLineWidth()

Permet de modifier l'épaisseur du trait.

Syntaxe	boolean pdf_setLineWidth(resource $objetPDF, double $largeur)

`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$largeur`	Largeur du trait (en points).
`retour`	`TRUE`.

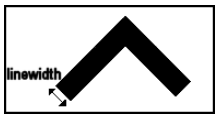


Figure 14.3 :
Largeur du trait

pdf_setLineJoin()

Définit la forme du coin formé par deux segments dont on a demandé à tracer le contour.

Syntaxe	`boolean pdf_setLineJoin(resource $objetPDF, int $formeCoin)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$formeCoin`	Au choix : `0` pour un coin obtus. `1` pour un coin arrondi. `2` pour un coin plat.
`retour`	`TRUE`.

Figure 14.4 :
Différents codes pour différents coins

pdf_setLineCap()

Définit la façon dont doivent être affichées les extrémités des segments.

Syntaxe	`boolean pdf_setLineCap(resource $objetPDF, int $style)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$style`	Au choix : `0` pour une extrémité plate. `1` pour une extrémité arrondie. `2` pour une extrémité carrée.
`retour`	`TRUE`.

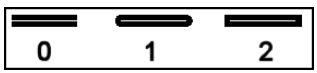


Figure 14.5 :
Différents codes pour différents bouts

pdf_setMiterLimit()

Définit la hauteur du coin (cf. dessin).

Syntaxe	`boolean pdf_setmiterlimit(resource $objetPDF, double $hauteur)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$hauteur`	Hauteur maximale du coin.
`retour`	`TRUE`.

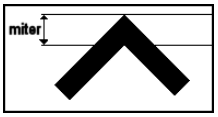


Figure 14.6 :
Hauteur d'un coin (miter)

Il est grand temps à présent de mettre en pratique certaines des fonctions vues jusque-là.

Listing 14.2 : pdf2.php

```
<?php
// Creation d'un pdf
$pdf = pdf_new();
// Ouverture d'un fichier en ecriture le nom
// du fichier est un chemin absolu.
pdf_open_file($pdf, "");
pdf_begin_page($pdf, 595, 842);
// Deplacement du curseur en (50, 740)
pdf_moveto($pdf, 50, 740);
// Definit un trait jusqu'en (315, 740)
pdf_lineto($pdf, 340, 740);
// Definition d'un arc de cercle dans le sens trigonometrique
pdf_arc ($pdf, 340, 700, 40, 90, 270);
// Definition d'un arc de cercle dans le sens horaire
pdf_arcn($pdf, 340, 620, 40, 90, 180);
// Deplacement du curseur en (50, 740)
pdf_moveto($pdf, 50, 740);
// Definition d'un cercle de centre (30, 740) et de rayon 20
pdf_circle($pdf, 30, 740, 20);
// Dessine le circuit trace precedemment et le remplit
pdf_fill($pdf);
//Modification de l'epaisseur du trait pour le trace suivant
pdf_setlinewidth($pdf, 10);
// Deplacement du curseur en (50, 400)
pdf_moveto($pdf, 50, 400);
// Definition d'un rectangle en (50, 600) de largeur 20
// et de hauteur 40
pdf_rect($pdf, 50, 600, 20, 40);
// Dessin du rectangle
pdf_stroke($pdf);
// Termine la page
```

```
pdf_end_page($pdf);
// Ferme le document PDF
pdf_close($pdf);
// Récupère les données du document PDF
$donnees = pdf_get_buffer($pdf);
// Efface l'objet PDF de la memoire et les ressources associées
pdf_delete($pdf);

/*************************************/
/* Cette partie concerne l'affichage */
/* du document, les données sont     */
/* prêtes, il suffit de les envoyer. */
/*************************************/
header("Content-type: application/pdf");
header("Content-Length: ".strlen($donnees));
header("Content-Disposition: inline; filename=test.pdf");
echo $donnees;
?>
```

Voici ce qui doit être obtenu :

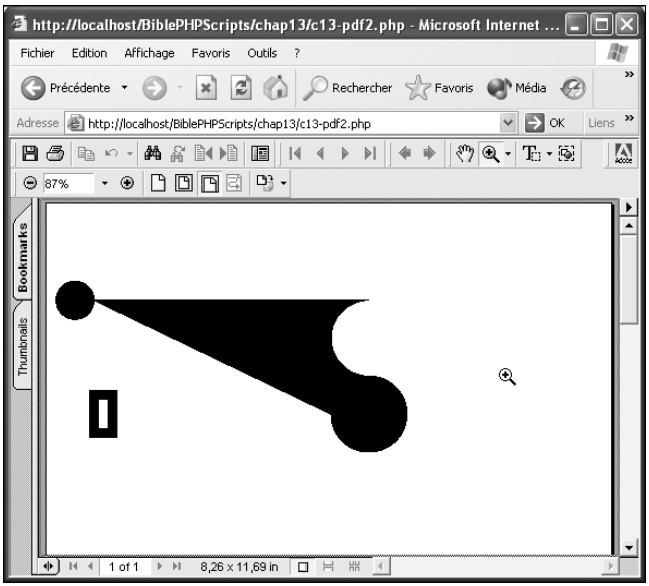


Figure 14.7 : *Dessiner un PDF en PHP*

Les pointillés

Il est possible de définir des pointillés simples...

pdf_setDash()

Permet de définir le type de pointillés.

Syntaxe	`void pdf_setDash(resource $objetPDF, double $trace, double $nonTrace)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$trace`	Nombre de points à tracer avant de suspendre le tracé.
`$nonTrace`	Nombre de points à ne pas tracer avant de reprendre le tracé.
`retour`	`TRUE`.

... comme de plus complexes.

pdf_setPolyDash()

Permet de définir un type de pointillés complexe.

Syntaxe	`boolean pdf_setPolyDash(resource $objetPDF, array $tableauPoints)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$tableauPoints`	Tableau de points (nombre de points à tracer, puis nombre de points à ne pas tracer, puis nombre de points à tracer, etc).
`retour`	`TRUE`.

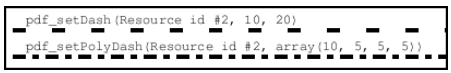


Figure 14.8 : *Les pointillés*

Réinitialiser les paramètres

Il est possible de rétablir les valeurs par défaut de ces paramètres avec la fonction suivante :

pdf_initGraphics()

Réinitialise les paramètres graphiques (couleurs, tracés, axes) à leurs valeurs par défaut.

Syntaxe	`boolean pdf_initGraphics(resource $objetPDF)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`retour`	`TRUE`.

Comme indiqué, cette fonction rétablit les axes de coordonnées à leur position initiale, car (nous le verrons plus loin dans ce chapitre) il est possible de les modifier.

14.8. Inclure une image

Pour inclure une image, vous devez au préalable ouvrir un fichier image, ce qui peut se faire avec `pdf_open_image_file()`. Les ressources ainsi allouées devront plus tard être libérées par un appel à `pdf_close_image()`.

pdf_open_image_file()

Lit une image depuis un fichier.

Syntaxe	`int pdf_open_image_file(resource $objetPDF, string $typeImage, string $nomFichier, string $parametreChaine, string $parametreEntier)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$typeImage`	Type de l'image, au choix "`jpeg`", "`tiff`", "`gif`" ou "`png`".
`$nomFichier`	Nom et chemin du fichier.
`$parametreChaine`	Au choix : `NULL`, `mask`, `masked`, `ignoremask`, `invert`, `page` ou `colorize`. Ce paramètre est optionnel avec PHP4.
`$parametreEntier`	`0`, l'identifiant de l'image du masque ou numéro de page. Ce paramètre est optionnel avec PHP4.
`retour`	Identifiant de l'image.

pdf_close_image()

Libère les ressources allouées par une image.

Syntaxe	`void pdf_close_image(resource $objetPDF, int $idImage)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$idImage`	Identifiant d'image tel que retourné par `pdf_open_image_file()`.

Pour ajouter l'image à la page, vous disposez de `pdf_place_image()`.

pdf_place_image()

Place une image dans la page.

Syntaxe	`boolean pdf_place_image(resource $objetPDF, int $idImage, double $x, double $y, double $echelle)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$idImage`	Identifiant d'image tel que retourné par `pdf_open_image_file()` ou `pdf_open_memory_image()` (ou identifiant de modèle tel que retourné par `pdf_begin_template()`).
`$x`	Abscisse du bord gauche de l'image.
`$y`	Ordonnée du bord inférieur de l'image.
`$echelle`	Échelle pour agrandir ou rétrécir l'image.
`retour`	`TRUE`.

Le script suivant ouvre une image dans une page PDF de 100 sur 100.

Listing 14.3 : imagefile.php

```
<?php
// Creation d'un pdf
$pdf = pdf_new();
pdf_open_file($pdf, "");
pdf_begin_page($pdf, 100, 100);

$logopdf = pdf_open_image_file($pdf, "png", "logo.png", "", 0);
pdf_place_image($pdf, $logopdf, 0, 0, 1);
pdf_close_image($pdf, $logopdf);

// Termine la page
pdf_end_page($pdf);
// Ferme le document PDF
pdf_close($pdf);
// Lit les données du document PDF
$donnees = pdf_get_buffer($pdf);
// Efface l'objet PDF de la memoire et les ressources associees
pdf_delete($pdf);

/*************************************/
/* Cette partie concerne l'affichage */
/* du document, les données sont     */
/* prêtes, il suffit de les envoyer. */
/*************************************/
header("Content-type: application/pdf");
header("Content-Length: ".strlen($donnees));
header("Content-Disposition: inline; filename=test.pdf");
echo $donnees;
?>
```

dont le résultat est :

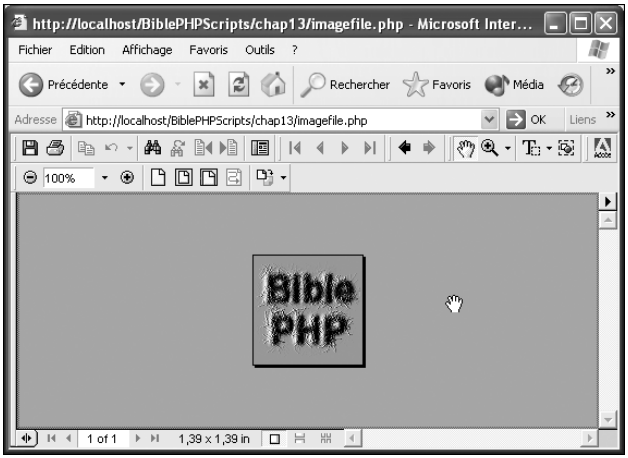


Figure 14.9 : *Insérer une image dans un document PDF*

Il est également possible d'utiliser des images ouvertes ou créées avec `GD`.

pdf_open_memory_image()

Ouvre une image obtenue par la librairie `GD`.

Syntaxe	`int pdf_open_memory_image(resource $objetPDF, resource $idImageGD)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$idImageGD`	Identifiant d'image tel que retourné par les fonctions `imageCreate()` ou `imageCreateFromXXX()` de la bibliothèque `GD`.
`retour`	Identifiant de l'image (PDF).

Découpe d'une image

Il est assez simple de ne représenter qu'une partie de l'image, ou de lui donner des contours ayant une forme un peu particulière. C'est ce que permet la fonction `pdf_clip()`.

pdf_clip()

Permet de découper une partie de l'image définie précédemment par un tracé.

Syntaxe	`boolean pdf_clip(int $objetPDF)`

`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`retour`	`TRUE`.

```
<?php
    // (...) Initialisation

    $img = pdf_open_image_file($pdf, "png", "logo.png", NULL, 0);
    pdf_circle($pdf, 50, 50, 50);
    pdf_clip($pdf);
    pdf_place_image($pdf, $img, 0, 0, 1);
    pdf_close_image($pdf, $img);

    // (...) Cloture
?>
```

Figure 14.10 :
Utilisation de clip

14.9. Ajouter des liens et des annotations

Liens PDF

Il vous est possible d'insérer un lien vers une autre page du document...

pdf_add_locallink()

Ajoute un lien vers une autre page du même fichier PDF.

Syntaxe	`boolean pdf_add_locallink(resource $objetPDF, double $x1, double $y1, double $x2, double $y2, int $noPage, string $zoom)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$x1`	Abscisse du point inférieur gauche.
`$y1`	Ordonnée du point inférieur gauche.
`$x2`	Abscisse du point supérieur droit.
`$y2`	Ordonnée du point supérieur droit.
`$noPage`	Page de destination.
`$zoom`	Au choix : "`retain`" pour conserver le facteur de zoom actuel à l'ouverture du document lié. "`fitpage`" pour adapter le zoom de façon à ce que le document lié tienne dans la fenêtre.

	"fitwith" pour adapter le zoom de façon à ce que la largeur du document lié tienne dans la fenêtre. "fitheight' pour adapter le zoom de façon à ce que la hauteur du document lié tienne dans la fenêtre. "fitbbox" pour adapter le zoom de façon à ce que le contenu du document (donc sans tenir compte des marges) tienne dans la fenêtre.
retour	TRUE.

... ou vers celle d'un autre document.

pdf_add_pdflink()

Ajoute un lien vers un fichier PDF.

Syntaxe	boolean pdf_add_pdflink(int $objetPDF, double $x1, double $y1, double $x2, double $y2, string $nomFichier, int $noPage, string $zoom)
$objetPDF	Identifiant tel que retourné par pdf_new().
$x1	Abscisse du point inférieur gauche.
$y1	Ordonnée du point inférieur gauche.
$x2	Abscisse du point supérieur droit.
$y2	Ordonnée du point supérieur droit.
$nomFichier	Nom du fichier vers lequel faire un lien.
$noPage	Numéro de la page à ouvrir.
$zoom	Au choix : "retain" pour conserver le facteur de zoom actuel à l'ouverture du document lié. "fitpage" pour adapter le zoom de façon à ce que le document lié tienne dans la fenêtre. "fitwith" pour adapter le zoom de façon à ce que la largeur du document lié tienne dans la fenêtre. "fitheight' pour adapter le zoom de façon à ce que la hauteur du document lié tienne dans la fenêtre. "fitbbox" pour adapter le zoom de façon à ce que le contenu du document (donc sans tenir compte des marges) tienne dans la fenêtre.
retour	TRUE.

Liens Internet

pdf_add_weblink()

Ajoute un lien vers une adresse Internet.

Syntaxe	`boolean pdf_add_weblink(resource $objetPDF, double $x1, double $y1, double $x2, double $y2, string $url)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$x1`	Abscisse du point inférieur gauche.
`$y1`	Ordonnée du point inférieur gauche.
`$x2`	Abscisse du point supérieur droit.
`$y2`	Ordonnée du point supérieur droit.
`$url`	Adresse du site web de destination.
`retour`	`TRUE`.

Liens vers un fichier

pdf_add_launchlink()

Ajoute un lien vers un fichier.

Syntaxe	`int pdf_add_launchlink(resource $objetPDF, double $x1, double $y1, double $x2, double $y2, string $nomFichier);`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$x1`	Abscisse du point inférieur gauche.
`$y1`	Ordonnée du point inférieur gauche.
`$x2`	Abscisse du point supérieur droit.
`$y2`	Ordonnée du point supérieur droit.
`retour`	`TRUE`.

Annotations

pdf_add_note()

Ajoute une annotation pour la page courante.

Syntaxe	`boolean pdf_add_note(resource $objetPDF, double $x1, double $y1, double $x2, double $y2, string $contenu, string $titre, string $icone, boolean $ouvrir)`

`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$x1`	Abscisse du point inférieur gauche de la zone de déploiement de l'annotation.
`$y1`	Ordonnée du point inférieur gauche de la zone de déploiement de l'annotation.
`$x2`	Abscisse du point supérieur droit de la zone de déploiement de l'annotation.
`$y2`	Ordonnée du point supérieur droit de la zone de déploiement de l'annotation.
`$contenu`	Texte de l'annotation.
`$titre`	Titre de l'annotation.
`$icone`	Icône a afficher, au choix : `"comment"`, `"insert"`, `"note"`, `"paragraph"`, `"newparagraph"`, `"key"` ou `"help"`.
`$ouvrir`	`TRUE` si le commentaire doit être visible par défaut, `FALSE` sinon.
`retour`	`TRUE`.

Icône
Sous Windows XP et Acrobat Reader 5.0, l'icône affichée est toujours note. Il semblerait que ce soit le cas pour toutes les versions 3 d'Acrobat Reader et au moins les versions 4 sous UNIX.
Selon la documentation officielle, cela aurait dû fonctionner sous l'environnement de test… Mais en pratique, cela ne fonctionne pas.

Modifier l'apparence des contours des liens et annotations

Il est possible de définir l'apparence des contours des liens et annotations grâce aux fonctions suivantes :

pfd_set_border_style()

Définit l'aspect des bordures (largeur du trait, pointillés ou non) des liens et des annotations.

Syntaxe	`boolean pdf_set_border_style(resource $objetPDF, string $style, double $epaisseur)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$style`	Au choix : `"solid"` pour un trait continu. `"dashed"` pour des pointillés.
`$epaisseur`	Épaisseur du trait.
`retour`	`TRUE`.

La nature des pointillés peut évidemment être précisée.

pdf_set_border_dash()

Permet de définir le type des pointillés appliqués aux contours des liens et des annotations.

Syntaxe	`boolean pdf_set_border_dash(resource $objetPDF, double $trace, double $nonTrace)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$trace`	Nombre de points à tracer avant de suspendre le tracé.
`$nonTrace`	Nombre de points à ne pas tracer avant de reprendre le tracé.
`retour`	`TRUE`.

pdf_set_border_color()

Définit la couleur des contours des liens et annotations.

Syntaxe	`boolean pdf_set_border_color(resource $objetPDF, double $rouge, $vert, $bleu)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$rouge`	Composante rouge de la couleur du contour (entre `0` et `1`).
`$vert`	Composante verte de la couleur du contour (entre `0` et `1`).
`$bleu`	Composante bleue de la couleur du contour (entre `0` et `1`).
`retour`	`TRUE`.

14.10. Ajouter des fichiers attachés et aperçus (thumbnails)

pdf_add_thumbnail()

Ajoute une petite image (thumbnail) pour la page courante.

Syntaxe	`boolean pdf_add_thumbnail(resource $objetPDF, int $idImage)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$idImage`	Identifiant d'image tel que retourné par `pdf_open_image_file()` ou `pdf_open_memory_image()`.
`retour`	`TRUE`.

pdf_attach_file()

Ajoute un fichier attaché pour la page courante.

Syntaxe	`boolean pdf_attach_file(int $objetPDF, double $x1, double $y1, double $x2, double $y2, string $nomFichier, string $description, string $auteur, string $mimetype, string $icone)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$x1`	Abscisse du point inférieur gauche.
`$y1`	Ordonnée du point inférieur gauche.
`$x2`	Abscisse du point supérieur droit.
`$y2`	Ordonnée du point supérieur droit.
`$nomFichier`	Nom du fichier à attacher.
`$description`	Description du fichier attaché.
`$auteur`	Auteur du fichier attaché.
`$mimetype`	Type MIME du fichier (ex: "text/plain").
`$icone`	Icône associée, au choix : "`graph`" graphique. "`paperclip`" trombone. "`pushpin`" punaise. "`tag`" étiquette.
`retour`	`TRUE`.

14.11. Modifier le système de coordonnées

pdf_translate()

Redéfinit l'origine du système.

Syntaxe	`boolean pdf_translate(resource $objetPDF, double $x, double $y)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$x`	Abscisse du nouveau centre.
`$y`	Ordonnée du nouveau centre.
`retour`	`TRUE`.

pdf_rotate()

Permet de tourner le système de coordonnées.

Syntaxe	`boolean pdf_rotate(resource $objetPDF, double $angle)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$angle`	Angle de rotation (en degrés) par rapport au système de coordonnées actuel.
`retour`	`TRUE`.

pdf_skew()

Modifie les axes du système de coordonnées (permet d'avoir un repère non orthonormé).

Syntaxe	`boolean pdf_skew(resource $objetPDF, double angle1, double angle2)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$angle1`	Angle de rotation (en degrés) de l'axe des abscisses par rapport au système de coordonnées actuel.
`$angle2`	Angle de rotation (en degrés) de l'axe des ordonnées par rapport au système de coordonnées actuel.
`retour`	`TRUE`.

pdf_scale()

Redéfinit l'échelle.

Syntaxe	`boolean pdf_scale(resource $objetPDF, double $echelleX, double $echelleY)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$echelleX`	Coefficient à appliquer sur l'échelle des abscisses.
`$echelleY`	Coefficient à appliquer sur l'échelle des ordonnées.
`retour`	`TRUE`.

Si par hasard vous ne trouviez pas votre bonheur avec les fonctions précédentes, vous pouvez encore préciser votre propre matrice de transformation (i.e. combiner les appels précédents en un appel unique).

pdf_setMatrix()

Remplace la matrice de transformation courante par une autre.

Syntaxe	`boolean pdf_setMatrix(resource $objetPDF, double $m11, double $m21, double $m12, double $m22, double $m13, double $m23)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$m11, $m21, $m12, $m22, $m13, $m23`	Les éléments de la matrice de transformation (3x2) `[[$m11, $m12, $m13][$m21, $m22, $m23]]`.
`retour`	`TRUE`.

pdf_concat()

Concatène une matrice à la matrice courante de transformation.

Syntaxe	`boolean pdf_concat(resource $objetPDF, double $m11, double $m21, double $m12, double $m22, double $m13, double $m23)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$m11, $m21, $m12, $m22, $m13, $m23`	Les éléments de la matrice de transformation (3x2) `[[$m11, $m12, $m13][$m21, $m22, $m23]]`.
`retour`	`TRUE`.

14.12. Lire, sauvegarder et restaurer les paramètres courants

pdf_get_value()

Récupère la valeur d'un paramètre (certains peuvent être modifiés par `pdf_set_value()`).

Syntaxe	`double pdf_get_value(int $objetPDF, string $cle, double $valeur)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$cle`	Nom de la valeur à récupérer.
`$valeur`	Nouvelle valeur. Ce paramètre est optionnel avec PHP4.
`retour`	La valeur recherchée.

Voici un tableau de différentes valeurs pouvant être obtenues :

Tableau 14.4 : Valeurs pouvant être obtenues par pdf_get_value()

Clé	Description
`major`	Récupère le numéro principal de version de `PDFLib`.
`minor`	Récupère le numéro secondaire de version de `PDFLib`.
`revision`	Récupère le numéro de révision de `PDFLib`.
`version`	Récupère le numéro de version complet sous la forme `major.minor.revision` suivi éventuellement de chaînes telles que `beta`, `rc`...
`scope`	Récupère le nom de l'étendue courante.
`pagewidth`	Largeur de la page courante en points.
`pageheight`	Hauteur de la page courante en points.

`Cropbox`
`BleedBox`
`ArtBox`
`TrimBox`
	Change les tailles des boîtes de la page courante. Le nom de la clé doit être suivi d'un slash `/` puis de `llx` (abscisse du point inférieur gauche), `lly` (ordonnée du point inférieur gauche), `urx` (abscisse du point supérieur gauche), `ury` (ordonnée du point supérieur droit). Par exemple `Cropbox/llx` permettra de modifier l'abscisse inférieure gauche de la Cropbox.

`font`	Retourne l'identifiant de la police d'écriture qui doit avoir été définie par `pdf_setfont()`.
`fontsize`	Retourne la taille de la police d'écriture précédemment définie par `pdf_setfont()`.

`capheight`
`ascender`
`descender`
	Retourne les valeurs des tailles pour l'affichage de caractères en fraction de taille de police.
	`Capheight` est la taille d'une majuscule.
	`Ascender` est la taille d'une lettre haute telle que l, f, k, t…
	`Capheight` est juste la taille de la partie basse d'une lettre descendante comme j, p, q…
	Après ce tableau se trouve une image expliquant ces trois valeurs.

`leading`	Modifie l'espace entre deux lignes de base de deux lignes consécutives.
`textrise`	Modifie l'espace entre la position désirée du texte et la ligne de base.
`textrendering`	`0` pour du texte plein. `1` pour n'avoir que le contour du texte. `2` pour du texte plein et le contour. `3` pour ne pas afficher le texte mais l'ajouter au chemin.
`horizscaling`	Pourcentage pour élargir ou rétrécir le texte.
`charspacing`	Espace entre deux caractères à ajouter à l'espacement par défaut.
`wordspacing`	Espace à ajouter à la taille du caractère d'espacement.
`textx`	Retourne l'abscisse courante du pointeur de texte.
`texty`	Retourne l'ordonnée courante du pointeur de texte.
`currentx`	L'abscisse du point graphique courant.
`currenty`	L'ordonnée du point graphique courant.

Clé	Description
`Imagewidth` `imageheight`	Retourne la largeur et la hauteur d'une image en pixel. Il faut passer en paramètre l'identifiant de l'image.
`resx` `resy`	Retourne les résolutions horizontales et verticales d'une image. Il faut passer en paramètre l'identifiant de l'image. Si la valeur est positive, la valeur retournée est exprimée en points par pouce ; sinon la valeur n'a pas de sens, mais permet de déterminer le rapport entre la largeur et la hauteur d'un point. Si la valeur est nulle, alors l'information sur la résolution n'est pas accessible.

Figure 14.11 : *capheight, ascender et descender en images*

pdf_set_parameter()

Permet de définir un paramètre.

Syntaxe	`void pdf_set_parameter(resource $objetPDF, string $cle, string $valeur)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$cle`	Nom du paramètre.
`$valeur`	Valeur du paramètre.

Voici un tableau des différents paramètres attribuables :

Tableau 14.5 : Paramètres attribuables par pdf_set_parameter()

Clé	Description	Défaut
`compatibility`	Permet de définir la compatibilité. Elle peut être définie en 1.3, 1.4 ou 1.5 pour Acrobat 3, Acrobat 4 ou Acrobat 5. Ce changement de paramètre doit être effectué avant tout appel à `pdf_open_*()`.	1.3
`Prefix`	Préfixe utilisé dans un fichier UPR.	
`Resourcefile`	Nom du fichier de configuration UPR.	
`Serial`	Définit le numéro de série de la bibliothèque `PDFLib` ; il ne peut être modifié qu'une seule fois avant le premier appel à `pdf_begin_page()`.	
`Warning`	Permet ou supprime les messages d'alerte. Peut être mis à "`true`" ou "`false`". (Par défaut, les messages sont apparents.)	"`true`"

Clé	Description	Défaut
FontAFM FontPFM FontOutline Encoding	Les fichiers de ressources tels qu'ils doivent apparaître dans un fichier UPR. Pour en ajouter plusieurs, il suffit d'appeler plusieurs fois la fonction.	
fontwarning	SI cette valeur est mise à "false" alors, pdf_findfont() renverra 0 au lieu d'un message d'erreur.	"true"
underline overline strikeout	Permet de définir l'aspect d'une chaîne de caractères ("true" ou "false") : Underline pour le soulignement. Overline pour le surlignage. Strikeout pour du texte barré.	"false"
native-unicode	Si la valeur est mise à "true" alors le texte Unicode natif est permis.	"false"
Fillrule	Change la règle de remplissage courante. Cette règle est utilisée par les visionneurs de fichier *PDF*, mais conduisent au même résultat. Les valeurs permises sont "winding" ou "evendd".	"winding"
Image-warning	Ce paramètre peut servir à savoir pourquoi une image n'a pas pu être ouverte correctement. Si ce paramètre est mis à "true" alors un message d'erreur est généré, sinon aucun message d'erreur n'est généré.	"false"
c	Contrôle de l'importation des attributs graphiques avec les pages et mises en page. "true", les attributs graphiques seront importés. "false", les attributs graphiques ne seront pas importés.	"true"
openaction	Définit l'action d'ouverture ou encore le grossissement de la première page du document. Les différentes valeurs possibles sont : "retain" pour conserver la valeur actuelle. "fitpage" pour que le zoom fasse en sorte que la page tienne à l'écran. "fitwidth" pour que la largeur tienne à l'écran. "fitheight" pour que la hauteur tienne à l'écran. "fitbbox" pour que le contenu de la page (sans tenir compte des bords) tienne à l'écran.	"retain"

Clé	Description	Défaut
openmode	Définit l'apparence quand le document est ouvert. "none", ni les signets ni les imagettes ne sont visibles. "bookmarks", les signets s'afficheront à l'ouverture du document. "thumbnails", les imagettes s'afficheront. "fullscreen", le document s'ouvrira sur tout l'écran si celui-ci n'est pas dans un navigateur Internet.	"bookmarks" si le document contient des signets. "none" sinon.
bookmark-dest	Définit le zoom pour l'affichage des signets. Les différentes valeurs possibles sont : "retain" pour conserver la valeur actuelle. "fitpage" pour que le zoom fasse en sorte que la page tienne à l'écran. "fitwidth" pour que la largeur tienne à l'écran. "fitheight" pour que la hauteur tienne à l'écran. "fitbbox" pour que le contenu de la page (sans tenir compte des bords) tienne à l'écran.	"retain"
transition	Définit la transition d'une page à une autre. Les différentes valeurs possibles sont : "split", "blinds", "box", "wipe", "dissolve", "glitter", "replace".	"replace"

Sauvegarder et restaurer

Il est également possible de sauvegarder l'environnement courant pour le réutiliser plus tard. Les axes de coordonnées, couleurs, épaisseurs de traits peuvent ainsi être changés après avoir sauvegardé l'environnement, pour ensuite recharger les mêmes paramètres de départ.

pdf_save()

Enregistre l'environnement courant.

Syntaxe	`boolean pdf_save(resource $objetPDF)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`retour`	`TRUE`.

pdf_restore()

Redéfinit l'environnement graphique avec les informations stockées lors de l'appel à `pdf_save()`.

Syntaxe	`boolean pdf_restore(resource $objetPDF)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`retour`	`TRUE`.

14.13. Créer un modèle

Il est possible de créer un modèle de page qui servira de base pour la création de pages. L'avantage principal réside dans le fait qu'une modification apportée à ce modèle s'appliquera à toutes les pages y faisant référence.

Après avoir créé un modèle, il suffit, pour l'utiliser,il de le considérer comme une image. Ainsi, pour l'ajouter à la page, vous n'aurez qu'à appeler la fonction `pdf_place_image()`.

Un modèle de page doit être défini en dehors de toute section page (`pdf_begin_page()` ...`pdf_end_page()`). Un modèle de page peut faire appel à un autre modèle de page, mais pas à lui-même.

Toutes les fonctions agissant sur le texte, les graphiques et les couleurs peuvent être utilisées, exception faite des fonctions d'ouverture et de fermeture d'images et des éléments hypertextes.

Introduire des images
Il n'est pas possible d'ouvrir des images dans les modèles, mais il est possible de les placer. Il suffit donc de les ouvrir avant de définir le modèle.

pdf_begin_template()

Débute la description d'un modèle.

Syntaxe	`int pdf_begin_template(resource $objetPDF, double $largeur, double $hauteur)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`$largeur`	Largeur du modèle.
`$hauteur`	Hauteur du modèle.
`retour`	Un identifiant de modèle.

pdf_end_template()

Termine la description d'un modèle.

Syntaxe	`boolean pdf_end_template(resource $objetPDF)`
`$objetPDF`	Identifiant tel que retourné par `pdf_new()`.
`retour`	`TRUE`.

L'exemple qui suit utilise ces deux fonctions.

Un exemple de script

L'exemple que nous avons choisi de détailler est un fax de facture avec une page de garde. Les renseignements nécessaires à la facture sont récupérés depuis un formulaire HTML.

Voici le script HTML du formulaire que nous ne détaillerons pas (il ne s'agit ni plus ni mois que d'une page HTML) :

Listing 14.4 : facture.html

```
<html>
  <head><title>Facture></title></head>
  <body>
    <center><h1>
      <font color="#0000FF">Creation de facture en fichier PDF</font>
    </h1></center>
    <form method="post" action="facture.php">
      <h2>Informations sur le destinataire</h2>
      <table align="center">
      <tr>
        <td>Nom:</td>
        <td><input type="text" name="nom" /></td>
      </tr>
      <tr>
        <td>Prenom:</td>
        <td><input type="text" name="prenom" /></td>
      </tr>
      <tr>
        <td>Numero de fax:</td>
        <td><input type="text" name="fax" /></td>
      </tr>
      </table>
      <h2>Informations sur la facture</h2>
      <table align="center">
      <tr>
        <td>Numero de facture:</td>
        <td><input type="text" name="no" /></td>
      </tr>
      <tr>
```

```
        <td>Montant:</td>
        <td><input type="text" name="montant" /></td>
      </tr>
      <tr>
        <td>Prestation:</td>
        <td><textarea name="prestation"></textarea></td>
      </tr>
      </table>
      <center><input type="submit" /></center>
    </form>
  </body>
</html>
```

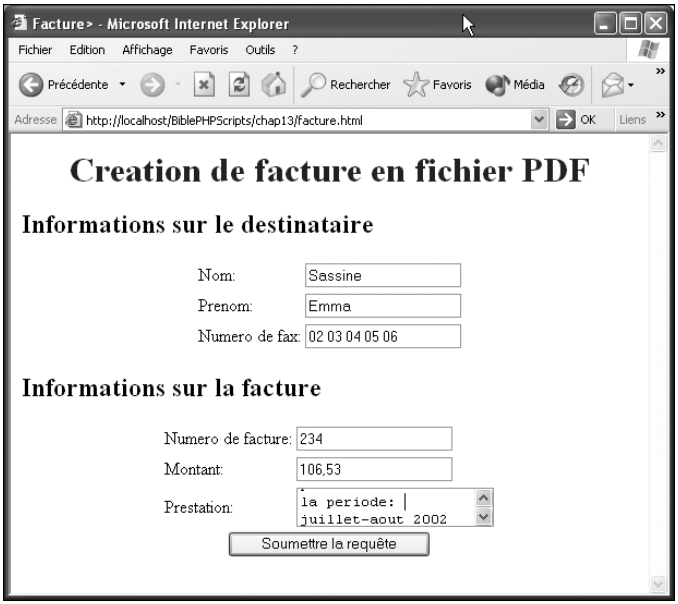


Figure 14.12 : *Exemple de formulaire rempli*

La page d'appel *facture.php* traite toute la partie créant le fichier PDF.

Listing 14.5 : facture.php

```
<?php
// Largeur de la page
$largeur=495;
// Hauteur de la page
$hauteur=742;

// Creation d'un pdf
$pdf = pdf_new();
// Allocation de memoire
pdf_open_file($pdf, "");
```

```
// Ouverture de l'image de logo
$logopdf = pdf_open_image_file($pdf, "png", "logo.png", "", 0);

// Definition d'un nouveau modele de taille inferieure a la page
$miseEnPage = pdf_begin_template($pdf, $largeur-100, $hauteur-100);

// Choisir la police Courier de taille 15
$font = pdf_findfont($pdf, "Courier", "host", FALSE);
if ($font) {
    pdf_setfont($pdf, $font, 15);
} else {
    die ("Police d'ecriture introuvable");
}
// Demande d'afficher que le contour du texte
pdf_set_value($pdf, "textrendering", 1);

// Affichage de "Bible PHP"
pdf_show_xy($pdf, "Bible PHP" , 175, $hauteur-125);

// Choisir la police Courier de taille 8
pdf_setfont($pdf, $font, 8);
// Demande d'afficher du texte plein
pdf_set_value($pdf, "textrendering", 0);

// Affichage de l'adresse de l'expediteur dans un rectangle
// en justifiant le texte
pdf_show_boxed($pdf,
               "Bible PHP\r\nChemin des acacias\r\n20876 Germantown\r\n".
               "Tel: 02 00 00 00 00", 0, $hauteur-100-100, 100, 100,
               "fulljustify");

// Placement du logo sur le modele
pdf_place_image($pdf, $logopdf, $largeur-100-50, $hauteur-100-50, 0.5);

//Changement de la taille du texte à 8 points
pdf_setfont($pdf, $font, 8);

// Affiche un texte centre en bas de page
pdf_show_boxed($pdf, "Copyright. SARL Capital 0 Euros", 0, 0,
               $largeur-100, 15, "center");

//Fin de la definition du modèle
pdf_end_template($pdf);

// Fermeture de l'image ouverte avant la declaration du modèle
pdf_close_image($pdf, $logopdf);


// Creer une nouvelle page: la page de garde
pdf_begin_page($pdf, $largeur, $hauteur);
// Placement du modele sur la page
```

```
pdf_place_image($pdf, $miseEnPage, 50, 50, 1);

// Choisir la police Courier de taille 10
$font = pdf_findfont($pdf, "Courier", "host", FALSE);
if ($font) {
    pdf_setfont($pdf, $font, 10);
} else {
    die("Police d'ecriture introuvable");
}
// Demande d'afficher du texte plein
pdf_set_value($pdf, "textrendering", 0);
// Affichage du nom et du numero de fax du destinataire
pdf_show_boxed($pdf, "Pour: ".$_POST["nom"]." ".$_POST["prenom"].
              "\r\nFax: ".$_POST["fax"], $largeur-200, $hauteur-300,
              150, 100, "right", "");

// Changement de la taille du texte à 14 points
pdf_setfont($pdf, $font, 14);
// Demande d'afficher du texte plein
pdf_set_value($pdf, "textrendering", 0);

// Texte à afficher
$texte = "Monsieur,\r\n\r\n".
         "    Vous trouverez dans ce fac-simile, une facture comme".
         " nous avions convenu. Le montant de cette facture est".
         " hors taxes et".
         " payable a l'ordre de \"Monsieur Bible PHP\".\r\n".
         "    Nous venons de modifier notre systeme de gestion de factures".
         " et nous serons heureux de vous envoyer cette facture par ".
         " courrier electronique dans un format".
         " lisible par tous, le PDF.\r\n\r\nRecevez Monsieur, mes meilleures".
         " salutations.\r\n\r\n                                                    ".
         "                                  Le president.";
// Affichage du texte precedent sur la page
pdf_show_boxed($pdf, $texte , 50, 100, $largeur-100, 400, "justify", "");

// Termine la page
pdf_end_page($pdf);

// Creer une nouvelle page: la facture
pdf_begin_page($pdf, $largeur, $hauteur);

// Placement du modele sur la page
pdf_place_image($pdf, $miseEnPage, 50, 50, 1);

// Choisir la police Courier de taille 10
$font = pdf_findfont($pdf, "Courier", "host", FALSE);
if ($font) {
    pdf_setfont($pdf, $font, 10);
} else {
    die("Police d'ecriture introuvable");
}
```

```
// Demande d'afficher du texte plein
pdf_set_value($pdf, "textrendering", 0);
// Affichage du nom et du numero de fax du destinataire
pdf_show_boxed($pdf, "Pour: ".$_POST["nom"]." ".$_POST["prenom"].
             "\r\nFax: ".$_POST["fax"], $largeur-200, $hauteur-300,
             150, 100, "right", "");
//Changement de la taille du texte a 35 points
pdf_setfont($pdf, $font, 35);

// Affichage du titre "Facture"
pdf_show_xy($pdf, "Facture" , 150, $hauteur-100-50);

//Changement de la taille du texte a 15 points
pdf_setfont($pdf, $font, 15);

// Texte à afficher
$texte="Facture No: ".$_POST["no"]."\r\nMontant: ".$_POST["montant"].
       " Euros H.T.\r\n".
       "Objet de prestation: ".stripslashes($_POST["prestation"])."\r\n".
       "TVA non applicable, article 293B du CGI";

// Affiche le texte aligne à gauche
pdf_show_boxed($pdf, $texte, 50, 100, $largeur-100, 300, "left", "");

// Termine la page
pdf_end_page($pdf);

// Ferme le document PDF
pdf_close($pdf);
// Récupère les données du document
$donnees = pdf_get_buffer($pdf);
// Efface l'objet PDF de la memoire et les ressources associees
pdf_delete($pdf);

/*************************************/
/* Cette partie concerne l'affichage */
/* du document, les données sont     */
/* prêtes, il suffit de les envoyer. */
/*************************************/
header("Content-type: application/pdf");
header("Content-Length: ".strlen($donnees));
header("Content-Disposition: inline; filename=test.pdf");
echo $donnees;
?>
```

Ce script crée un fichier PDF de deux pages ; la première est la page de garde d'un fax :

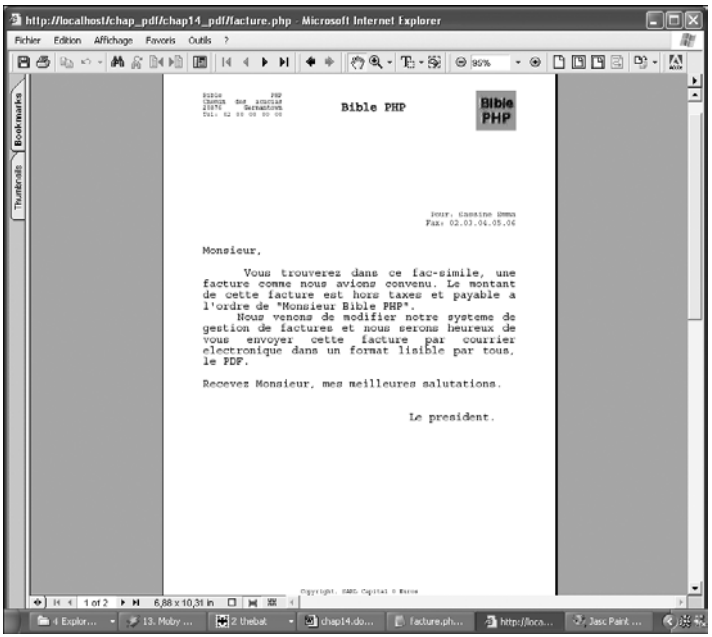


Figure 14.13 : *Page de garde*

La seconde page est la page de facture :

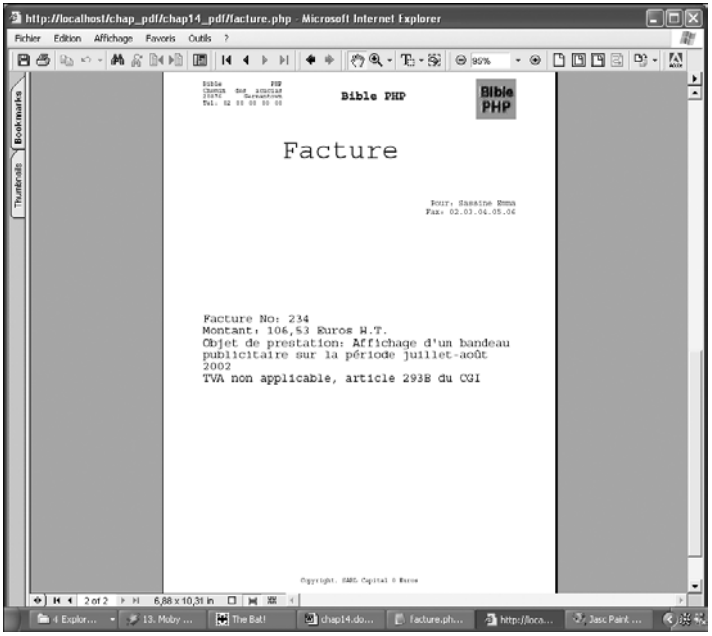


Figure 14.14 : *Page de facture*

Chapitre 15

L'utilisation de XML

15.1. Introduction

Présentation du langage XML

Ces dernières années, vous avez certainement beaucoup entendu parler du langage XML. La plupart en disent le plus grand bien, et il est de plus en plus utilisé dans l'industrie. Toutefois, beaucoup n'ont encore qu'une vague idée de ce qu'il est, tandis que d'autres pensent (à tort) qu'il s'agit d'une évolution du langage HTML.

En fait, XML définit avant tout des règles très simples de description d'informations (généralement stockées dans des fichiers). En quelques mots, respecter le standard XML, c'est stocker l'information sous forme de balises et d'attributs (un peu comme avec HTML).

En revanche, en aucun cas XML ne s'attache à décrire la façon dont les informations seront présentées. Il se peut d'ailleurs que les informations ne soient jamais affichées ou imprimées, mais seulement stockées et échangées entre serveurs et applications. Le traitement de l'information (y compris affichage ou impression) est laissé à la charge du parseur (ou analyseur) XML (logiciel pouvant être écrit en n'importe quel langage, y compris PHP, et qui pourra éventuellement s'appuyer sur le langage XSL).

Contrairement au langage HTML, XML ne définit ni balises ni attributs. C'est à l'utilisateur, selon son corps de métier et ses besoins, de les définir.

La bibliothèque libXml a fait sont entrée avec PHP5. Elle remplace avantageusement les autres précédemment utilisées tout en gardant les mêmes signatures de fonctions. Ainsi une application utilisant XML écrite pour PHP4 doit continuer à fonctionner sans problème en passant à PHP5 même si derrière il ne se passe pas exactement la même chose.

L'avantage de libXML est qu'elle supporte l'analyse SAX et DOM, de plus il se trouve que cette bibliothèque est très performante.


INTERNET

Comparatif
A titre d'information voici une adresse où vous pouvez comparer les différents analyseurs syntaxiques, vous verrez que libXml est très bien placée.
http://xmlbench.sourceforge.net/index.php?page=results.php


REMARQUE

Spécifications
Les langages XML et XSL sont décrits par le W3C.

Le format XML

Un fichier bien formé

Un fichier XML :

- Commence par un en-tête `<?xml version="1.0" encoding="UTF-8"?>` précisant qu'il s'agit d'un document XML respectant la norme de la version 1.0, et utilisant des caractères codés en "UTF-8" (il est possible de préciser un autre encodage).

- Ne possède qu'une balise racine (autrement dit, l'ensemble des balises du document XML, hormis l'en-tête doit être compris entre `<baliseracine>` et `</baliseracine>`). Peu importe le nom de cette balise.
- Toute balise ouverte doit être refermée (si il n'y a pas de balise ou de texte entre les balises ouvrante et fermante, `<balise></balise>` peut être remplacé par `<balise />`).
- Les noms des balises doivent commencer par une lettre ou un underscore ('_'). Les caractères suivants peuvent être des chiffres, des lettres, un underscore, un point ou un tiret.
- Un nom de balise ne peut pas commencer par "`xml`".
- Les balises ne peuvent se chevaucher (il ne peut y avoir `<balise1><balise2>` `</balise1></balise2>`).

Les noms des balises sont généralement en minuscules.

Un document respectant ces règles est dit *bien formé*.

Un fichier valide (DTD)

Bien qu'il suffise qu'un document soit bien formé pour qu'il s'agisse d'un document XML, ce ne sont généralement pas des conditions suffisantes pour assurer son bon traitement par un analyseur.

En effet, selon le corps de métier, nous nous attendrons à avoir un document XML contenant telle ou telle balise. Les contraintes seront certainement même plus fortes, puisqu'une balise donnée devra contenir un certain type d'attributs et de balises.

Pour assurer qu'un document respecte ces contraintes supplémentaires, ces dernières seront exprimées sous la forme d'un document appelé DTD, dont voici un exemple :

```
<?xml version="1.0" encoding="UTF-8" ?>
<!ELEMENT annuaire (personne*)>
<!ELEMENT personne (nom, personne, email+)>
<!ATTLIST personne profession (etudiant | professeur | chanteur | musicien)
✂ "etudiant">
<!ELEMENT nom (#PCDATA)>
<!ELEMENT prenom (#PCDATA)>
<!ELEMENT email (#PCDATA)>
```

Un tel document précise que la balise "`annuaire`" doit comporter entre zéro et plusieurs balises "`personne`" (ceci est indiqué par le caractère '`*`'). La balise "`personne`" doit contenir, dans cet ordre, exactement une balise "`nom`", une balise "`personne`" et une ou plusieurs balises "`email`" (ceci est indiqué par le caractère '`+`'). La balise "`personne`" possède un attribut "`profession`" qui pourra prendre une des valeurs parmi "`etudiant`", "`professeur`", "`chanteur`", "`musicien`". Si cet attribut n'est pas précisé, il prendra pour valeur "`etudiant`". Les balises "`nom`", "`prenom`" et "`email`" contiendront simplement du texte.

Nous n'irons pas plus loin dans la description de la DTD.

Un fichier DTD présente tout de même un gros défaut. En effet, il ne s'agit pas d'un document XML (bien formé). Ainsi, petit à petit, l'utilisation de la DTD devrait laisser place à son équivalent XML : "XML Schema" que nous n'aborderons pas ici.


INTERNET

XML, XML Schema et DTD
Vous trouverez plus d'informations sur XML, XML Schema et la DTD, sur les sites :
http://www.xmlfacile.com
http://www.xmlfr.org/documentations/
Sans oublier la référence (en anglais) :
http://www.w3.org/XML/
et sa traduction :
http://babel.alis.com/web_ml/xml/REC-xml.fr.html

Exemple de document XML

Comme nous l'avons vu, XML impose très peu de règles. Il peut donc convenir à tous les domaines d'applications. Prenons l'exemple d'une médiathèque désireuse de décrire sa liste de CD audio dans un document XML. Un tel fichier pourra ressembler à :

Listing 15.1 : mediatheque_01.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<mediatheque>
    <cdaudio interprete="Noir Désir" titre="des Visages des Figures">
        <chanson>
            <titre>L'enfant roi</titre>
            <duree>6:03</duree>
        </chanson>
        <chanson>
            <titre>Le grand incendie</titre>
            <duree>4:37</duree>
        </chanson>
        <chanson>
            <titre>Le vent nous portera</titre>
            <duree>4:48</duree>
        </chanson>
    </cdaudio>
    <cdaudio interprete="Placebo" titre="Without You I'm Nothing">
        <chanson>
            <titre>Pure Morning</titre>
            <duree>4:15</duree>
        </chanson>
        <chanson>
            <titre>Without You I'm Nothing</titre>
            <duree>4:30</duree>
        </chanson>
        <chanson>
            <titre>Every You, Every Me</titre>
            <duree>5:00</duree>
        </chanson>
    </cdaudio>
</mediatheque>
```

Mais, encore une fois, le choix n'étant pas imposé, il est fait de façon totalement arbitraire. Par exemple : nous aurions pu choisir de créer une balise "`interprete`" plutôt que d'en faire un attribut de la balise "`cdaudio`". Tout comme nous aurions pu avoir un attribut "`titre`" pour "`chanson`" plutôt que d'en faire une balise (ce qui aurait certainement été plus judicieux, mais aurait constitué un exemple moins intéressant).

Quoi qu'il en soit, nous avons (pour les parties renseignées) toutes les informations nécessaires au bon déroulement des opérations. Peut-être même que certaines de ces informations ne seront finalement pas utilisées (ou seulement dans certains cas).

Vous remarquerez qu'il est souhaitable d'indiquer que le document est un document XML répondant à la norme établie dans la version 1.0. Il est également préférable de préciser quel type d'encodage est utilisé : ISO-8859-1, UTF-8 (i.e. comment sont codés les caractères accentués ou les caractères non européens).


REMARQUE

Exemples d'encodage
UTF-8 : est destiné à supporter tous les styles de caractères.
US-ASCII : correspond aux caractères anglais.
ISO-8859-1 : correspond aux caractères utilisés en Europe de l'Ouest.
ISO-8859-2 : correspond aux caractères utilisés en Europe centrale.
ISO-8859-9 : correspond aux caractères turcs.
EUC-JP : correspond aux caractères japonais (sous UNIX).
etc.
Bref, il existe un grand nombre d'encodages. Mais, heureusement, un effort d'harmonisation a été entrepris et l'encodage UTF-8 (ou l'UTF-16) devrait, à terme, s'imposer.

Toutefois, les fichiers XML peuvent être un poil plus complexes s'ils utilisent, par exemple, des déclarations de type de documents `<!DOCTYPE ...>` incluant (pourquoi pas) des `<!ENTITY ...>` avec des références internes et externes. Le but étant d'utiliser des alias afin de remplacer, par exemple, un mot-clé par une longue chaîne de caractères ou par un fichier complet.

Listing 15.2 : mediatheque_01b.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE mediatheque [
    <!ENTITY OKComputer SYSTEM "mediatheque_02.xml">
    <!ENTITY nc "Non communiqué">
]>
<mediatheque>
    <cdaudio interprete="Noir Désir" titre="des Visages des Figures">
        <chanson>
            <titre>L'enfant roi</titre>
            <duree>6:03</duree>
        </chanson>
        <chanson>
            <titre>Le grand incendie</titre>
            <duree>4:37</duree>
        </chanson>
        <chanson>
```

```
            <titre>Le vent nous portera</titre>
            <duree>4:48</duree>
        </chanson>
    </cdaudio>
    <cdaudio interprete="Placebo" titre="Without You I'm Nothing">
        <chanson>
            <titre>Pure Morning</titre>
            <duree>4:15</duree>
        </chanson>
        <chanson>
            <titre>Without You I'm Nothing</titre>
            <duree>4:30</duree>
        </chanson>
        <chanson>
            <titre>Every You, Every Me</titre>
            <duree>5:00</duree>
        </chanson>
    </cdaudio>
    &OKComputer;
    <cdaudio interprete="Muse" titre="Showbiz">
        <chanson>
            <titre>Feeling Good</titre>
            <duree>&nc;</duree>
        </chanson>
    </cdaudio>
</mediatheque>
```

Listing 15.3 : mediatheque_02.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
    <cdaudio interprete="Radiohead" titre="OK Computer">
        <chanson>
            <titre>Karma Police</titre>
            <duree>4:23</duree>
        </chanson>
    </cdaudio>
```

Mais, même ce genre de fichier XML (faisant appel à un autre fichier XML) pourra être analysé par PHP, comme nous allons très bientôt le voir…

Utilisation des documents XML

Les documents XML pourront être lus par un analyseur XML (parseur), transformés par un analyseur XSL, ou pourront simplement servir de support d'échange (comme WDDX).

15.2. Installation

Depuis PHP 4, le support de XML est activé par défaut (mais peut être désactivé par l'option de compilation `--disable-xml`). Vous n'aurez donc, a priori, rien à faire de spécial pour pouvoir profiter des fonctions décrites dans ces chapitres. Depuis PHP5, une seule bibliothèque (libXml) permet de s'occuper des différentes méthodes d'analyse syntaxique et de transformation, il est cependant possible de désactiver certaines parties.

Vous pouvez vous assurer du support de libXML en appelant un script contenant uniquement `<?php phpinfo(); ?>`, qui devra alors afficher :

libxml	
libXML support	active
libXML Version	2.5.11
libXML streams	enabled

Figure 15.1 : *phpinfo()*

15.3. Les parseurs

Dans PHP4, le parseur appelé *expat* était intégré par défaut et il n'y avait rien à faire pour profiter de ses fonctions. Depuis PHP5, expat a été remplacé par libXml qui est également intégrée et qui offre en outre la même API. Ainsi les fonctions décrites ci-dessous pour libXML (PHP5) sont valides pour expat (PHP4) et inversement…

Un autre parseur appelé SimpleXML est disponible depuis PHP5, comme son nom l'indique son utilisation est très simple.

Le parseur SAX

Cette analyse de document est une analyse par événement. C'est-à-dire qu'à chaque fois qu'une situation particulière est rencontrée (ouverture d'une balise, fermeture d'une balise, texte contenu dans une balise, etc.), une fonction donnée est appelée.

Notre script PHP devra donc comporter :

- Les déclarations des fonctions devant être appelées pour les événements nous intéressant.

et les étapes suivantes :

- Création du parseur XML ;
- Pour chaque type d'événement, déclaration des fonctions auprès du parseur ;
- Lancement de l'analyse proprement dite ;
- Libération des ressources monopolisées par le parseur.

xml_parser_create

La création du parseur XML se fait par simple appel à la fonction `xml_parser_create()`.

xml_parser_create()

Initialisation d'un parseur XML.

Syntaxe	`resource xml_parser_create([string $encodage])`
`$encodage`	Argument optionnel (insensible à la casse) précisant le type d'encodage de la source parmi : `ISO-8859-1` (par défaut). `US-ASCII`. `UTF-8`.
`retour`	En cas de succès, une référence de parseur est retournée, `FALSE` sinon.

La référence retournée sera alors utilisée dans tous les appels aux fonctions XML (nous aurions préféré un bel objet avec une série de méthodes mais bon...).

Gestion des balises ouvrantes et fermantes avec xml_set_element_handler

Dans une version minimaliste (pour commencer) de l'analyseur, nous ne nous intéresserons qu'aux balises ouvrantes ou fermantes. Lorsque de tels éléments sont rencontrés, le parseur fait appel aux fonctions qui ont été préalablement référencées par `xml_set_element_handler()`.

xml_set_element_handler()

Déclare auprès du parseur les fonctions à appeler lorsqu'une balise ouvrante ou fermante est rencontrée.

Syntaxe	`boolean xml_set_element_handler(resource $parseur, string $fonctionBaliseOuvrante, string $fonctionBaliseFermante)`
`$parseur`	Référence du parseur telle que retournée par la fonction `xml_parser_create()`.
`$fonctionBaliseOuvrante`	Nom de la fonction en charge des balises ouvrantes.
`$fonctionBaliseFermante`	Nom de la fonction en charge des balises fermantes.
`retour`	`TRUE` si l'opération s'est déroulée correctement, `NULL` sinon.

Fonction balise ouvrante

La fonction (dont le nom sera précisé par `$fonctionBaliseOuvrante`) devant traiter les balises ouvrantes aura l'interface suivante :

Syntaxe	`[boolean] maFonction(resource $parseur, string $nomBalise, array $tableauAttributs)`

`$parseur`	Référence du parseur telle que retournée par la fonction `xml_parser_create()`.
`$nomBalise`	Nom de la balise ouvrante.
`$tableauAttributs`	Tableau associatif ayant pour clés les noms des attributs (en majuscules) contenus dans la balise et pour valeurs les valeurs des attributs.
`retour`	`TRUE` en cas de succès, `FALSE` sinon. (Ceci n'est pas requis ni pris en compte à ce jour par le parseur, mais c'est conseillé puisque cet état de fait peut évoluer).

L'ensemble des paramètres étant fourni par le parseur, la fonction aura accès à toutes les informations nécessaires au bon traitement des données du fichier XML.

Fonction balise fermante

La fonction (dont le nom sera précisé par `$fonctionBaliseFermante`) devant traiter les balises fermantes aura l'interface suivante :

Syntaxe	`[boolean] maFonction(resource $parseur, string $nomBalise)`
`$parseur`	Référence du parseur telle que retournée par la fonction `xml_parser_create()`.
`$nomBalise`	Nom de la balise fermante.
`retour`	`TRUE` en cas de succès, `FALSE` sinon. (Ceci n'est pas requis ni pris en compte à ce jour par le parseur, mais c'est conseillé puisque cet état de fait peut évoluer).

L'ensemble des paramètres étant fourni par le parseur, la fonction aura accès à toutes les informations nécessaires au bon traitement des données du fichier XML.

xml_parse

L'analyse d'un document XML consiste généralement à lire un fichier ligne par ligne et, pour chacune des lignes, à appeler `xml_parse()`, en précisant s'il s'agit ou non de la dernière ligne.

xml_parse()

Analyse une chaîne de caractères avec un parseur XML.

Syntaxe	`boolean xml_parse(resource $parseur, string $ligne [, boolean $derniereLigne])`
`$parseur`	Référence du parseur telle que retournée par la fonction `xml_parser_create()`.
`$ligne`	Chaîne de caractères à analyser.
`$derniereLigne`	`TRUE` s'il s'agit des dernières données à traiter, `FALSE` (par défaut) sinon.

retour	1 (mais pas strictement TRUE) si l'opération s'est déroulée correctement, 0 (mais pas strictement FALSE) en cas d'échec. Pour certaines fonctions de gestion des événements (xml_set_external_entity_ref_handler()), l'opération sera considérée en échec si elles ne retournent pas TRUE. Ce n'est en revanche pas le cas des fonctions xml_set_element_handler() et xml_set_character_data_handler().

Selon les éléments rencontrés, les différentes fonctions chargées des événements seront appelées.

xml_parser_free

À la fin de l'analyse, il est bon de libérer les ressources qui ont été allouées par le parseur. Pour cela, il suffit d'un simple appel à xml_parser_free().

xml_parser_free()

Libère les ressources allouées au parseur.

Syntaxe	boolean xml_parser_free(resource $parseur)
$parseur	Référence du parseur telle que retournée par la fonction xml_parser_create().
retour	TRUE en cas de succès, NULL en cas d'échec.

Premier exemple d'application

Ceci nous permet donc de créer notre premier script d'analyse de document XML (qui, pour l'instant, reste basique et ne gère pas les erreurs).

Listing 15.4 : expat_01.xml

```
<?php
$fichier = "../src_xml/mediatheque_01.xml";

// Exemple de fonction gérant
// les balises ouvrantes
function fonctionBaliseOuvrante($parseur, $nomBalise, $tableauAttributs)
{
    echo "La balise ouvrante : $nomBalise<br />\n";
    if ( count($tableauAttributs) > 0 ) {
        echo "...avec les attributs:<br />\n";
        while ( list($attribut, $valeur) = each($tableauAttributs) ) {
            echo "   $attribut = $valeur<br />\n";
        }
    }
```

```
    return TRUE;
}

// Exemple de fonction gérant
// les balises fermantes
function fonctionBaliseFermante($parseur, $nomBalise)
{
    echo "La balise fermante: $nomBalise<br />\n";

    return TRUE;
}

// Creation du parseur XML
$parseurXML = xml_parser_create();

// Association des fonctions
// de traitement des balises ouvrantes et fermantes
xml_set_element_handler($parseurXML, "fonctionBaliseOuvrante",
✂ "fonctionBaliseFermante");

// Ouverture du fichier
( $fp = fopen($fichier, "r") ) or
    die("Impossible d'ouvrir le fichier XML");

// Lecture du fichier ligne par ligne
echo "Lors du parcours du fichier XML. J'ai rencontré:<br />";
while ( $ligneXML = fgets($fp, 1024) ) {
    xml_parse($parseurXML, $ligneXML, feof($fp) ) or
        die("Erreur XML");
}

// Libération des ressources
xml_parser_free($parseurXML);
fclose($fp);
?>
```

Ceci aura pour effet d'afficher :

```
Lors du parcours du fichier XML. J'ai rencontré:
La balise ouvrante : MEDIATHEQUE
La balise ouvrante : CDAUDIO
...avec les attributs:
INTERPRETE = Noir Désir
TITRE = des Visages des Figures
La balise ouvrante : CHANSON
La balise ouvrante : TITRE
La balise fermante: TITRE
La balise ouvrante : DUREE
La balise fermante: DUREE
La balise fermante: CHANSON
La balise ouvrante : CHANSON
La balise ouvrante : TITRE
```

```
La balise fermante: TITRE
La balise ouvrante : DUREE
La balise fermante: DUREE
La balise fermante: CHANSON
La balise ouvrante : CHANSON
La balise ouvrante : TITRE
La balise fermante: TITRE
La balise ouvrante : DUREE
La balise fermante: DUREE
La balise fermante: CHANSON
La balise fermante: CDAUDIO
La balise ouvrante : CDAUDIO
...avec les attributs:
INTERPRETE = Placebo
TITRE = Without You I'm Nothing
La balise ouvrante : CHANSON
La balise ouvrante : TITRE
La balise fermante: TITRE
La balise ouvrante : DUREE
La balise fermante: DUREE
La balise fermante: CHANSON
La balise ouvrante : CHANSON
La balise ouvrante : TITRE
La balise fermante: TITRE
La balise ouvrante : DUREE
La balise fermante: DUREE
La balise fermante: CHANSON
La balise ouvrante : CHANSON
La balise ouvrante : TITRE
La balise fermante: TITRE
La balise ouvrante : DUREE
La balise fermante: DUREE
La balise fermante: CHANSON
La balise fermante: CDAUDIO
La balise fermante: MEDIATHEQUE
```

Première constatation : le document a été correctement analysé (on retrouve tous les attributs et balises contenus dans le fichier XML). Et c'est plutôt une bonne nouvelle !

Seconde constatation : tous les noms de balises et attributs ont été passés en majuscules.

Enfin, dans cet exemple, le contenu des balises n'a pas été traité. Ceci n'a rien d'étonnant, puisque nous nous sommes contentés de définir les fonctions des événements balises ouvrantes et balises fermantes.

La transformation ou non des noms des balises et des attributs est une option du parseur qui peut être fixée par la fonction `xml_parser_set_option()`.

xml_parser_set_option()

Fixe les options du parseur (respect de la casse, encodage de sortie).

Syntaxe	`boolean xml_parser_set_option(resource $parseur, int $option, mixed $valeur)`
`$parseur`	Référence du parseur telle que retournée par la fonction `xml_parser_create()`.
`$option`	Au choix, une valeur parmi : `XML_OPTION_CASE_FOLDING` : force le passage en majuscules (par défaut `TRUE`). `XML_OPTION_TARGET_ENCODING` : détermine l'encodage de sortie (par défaut, celui précisé du document en entrée).
`$valeur`	Valeur à donner à l'option.
`retour`	`TRUE` en cas de succès, rien (mais pas strictement `FALSE`) sinon.

Il est possible de vérifier la valeur d'une option avec `xml_parser_get_option()`.

xml_parser_get_option()

Retourne les options du parseur.

Syntaxe	`mixed xml_parser_get_option(resource $parseur, int $option)`
`$parseur`	Référence du parseur telle que retournée par la fonction `xml_parser_create()`.
`$option`	Au choix, une valeur parmi : `XML_OPTION_CASE_FOLDING` indiquant si les noms sont convertis en majuscules. `XML_OPTION_TARGET_ENCODING` indiquant l'encodage de sortie.
`retour`	Valeur de l'option.

Ce premier exemple nous a permis de bien comprendre comment se déroule l'analyse. Voici maintenant un exemple d'utilisation un peu plus concret, destiné à afficher la liste des interprètes et titres.

Listing 15.5 : expat_02.php

```
<?php
$fichier = "../src_xml/mediatheque_01.xml";

// Exemple de fonction gérant
// les balises ouvrantes
function fonctionBaliseOuvrante($parseur, $nomBalise, $tableauAttributs)
{
```

```
    switch ($nomBalise) {
        case "cdaudio" :
            echo $tableauAttributs["interprete"];
            echo " : ";
            echo $tableauAttributs["titre"];
            echo "<br />";
            break;
    }

    return TRUE;
}

// Exemple de fonction gérant
// les balises fermantes
function fonctionBaliseFermante($parseur, $nomBalise)
{
    // Dans ce cas, je n'ai rien à faire..

    return TRUE;
}

// Creation du parseur XML
$parseurXML = xml_parser_create();

// Force le parseur à respecter la casse
xml_parser_set_option($parseurXML, XML_OPTION_CASE_FOLDING, FALSE);

// Déclaration des fonctions de traitement
// des balises ouvrantes et fermantes
xml_set_element_handler($parseurXML, "fonctionBaliseOuvrante",
                                     "fonctionBaliseFermante");

// Ouverture du fichier
( $fp = fopen($fichier, "r") ) or
    die("Impossible d'ouvrir le fichier XML");

// Lecture du fichier ligne par ligne
while ( $ligneXML = fgets($fp, 1024) ) {
    xml_parse($parseurXML, $ligneXML, feof($fp) ) or
        die("Erreur XML");
}

// Libération des ressources
xml_parser_free($parseurXML);
fclose($fp);
?>
```

donnera effectivement :

```
Noir Désir : des Visages des Figures
Placebo : Without You I'm Nothing
```

Voyons maintenant le moyen de traiter les autres événements.

Gestion du texte contenu dans les balises avec xml_set_character_data_handler

La fonction `xml_set_character_data_handler()` vous permet de préciser quelle fonction doit traiter le texte contenu entre les balises ouvrante et fermante.

xml_set_character_data_handler()

Déclare auprès du parseur la fonction à appeler lorsque du texte est rencontré.

Syntaxe	`boolean xml_set_character_data_handler(resource $parseur, string $fonctionTexte)`
`$parseur`	Référence du parseur telle que retournée par la fonction `xml_parser_create()`.
`$fonctionTexte`	Nom de la fonction en charge du traitement du texte.
`retour`	`TRUE` si l'opération s'est déroulée correctement, `NULL` sinon.

Fonction texte des balises

La fonction (dont le nom sera précisé par `$fonctionTexte`) devant traiter le texte contenu entre les balises aura l'interface suivante :

Syntaxe	`[boolean] maFonction(resource $parseur, string $texte)`
`$parseur`	Référence du parseur telle que retournée par la fonction `xml_parser_create()`.
`$texte`	Le texte rencontré.
`retour`	`TRUE` en cas de succès, `FALSE` sinon. (Ceci n'est pas requis ni pris en compte à ce jour par le parseur, mais c'est conseillé puisque cet état de fait peut évoluer).

L'ensemble des paramètres étant fourni par le parseur, la fonction aura accès à toutes les informations nécessaires au bon traitement des données.

Gestion du caractère &

Lorsqu'une chaîne de caractères contient un & (indicateur de référence interne ou externe) celle-ci est scindée en plusieurs morceaux. Il y a donc dans ce cas plusieurs appels à la fonction décrite ici.

Nous allons donc pouvoir compléter notre exemple afin d'afficher les noms et durées des chansons disponibles sur les albums.

Listing 15.6 : expat_03.php

```
<?php
$fichier = "../src_xml/mediatheque_01.xml";

// Exemple de fonction gérant
// les balises ouvrantes
function fonctionBaliseOuvrante($parseur, $nomBalise, $tableauAttributs)
{
    // Pour les besoins de la fonction
    // fonctionTexte nous devons mémoriser
    // les noms des balises rencontrées
    // dans une variable globale
    global $balises;

    $balises[] = $nomBalise;

    switch ($nomBalise) {
        case "cdaudio" :
            echo "<b>";
            echo $tableauAttributs["interprete"];
            echo " : ";
            echo $tableauAttributs["titre"];
            echo "</b><br />";
            break;
    }

    return TRUE;
}

// Exemple de fonction gérant
// les balises fermantes
function fonctionBaliseFermante($parseur, $nomBalise)
{
    // Une fois sorti de la balise
    // il faut mettre à jour la variable
    // globale mémorisant les noms de balises
    // rencontrées, en supprimant le dernier nom
    global $balises;

    array_pop($balises);

    return TRUE;
}

function fonctionTexte($parseur, $texte)
{
    // Pour savoir a quelle balise se rapporte
    // ce texte, il faut avoir au préalable
    // mémorisé les balises rencontrées
    // dans une variable globale
    global $balises;
```

```
    // Toutefois, dans notre cas,
    // Nous ne nous interessons qu'à la dernière
    // balise rencontrée (sans tenir compte
    // des balises parentes)
    $derniereBalise = $balises[count($balises)-1];
    switch ($dernieBalise) {
        case "titre" :
             echo "- $texte";
             break;
        case "duree":
             echo " $texte<br />";
             break;
    }

    return TRUE;
}

// Creation du parseur XML
$parseurXML = xml_parser_create();

// Force le parseur a respecter la casse
xml_parser_set_option($parseurXML, XML_OPTION_CASE_FOLDING, FALSE);

// Déclaration des fonctions de traitement
// des balises ouvrantes et fermantes
xml_set_element_handler($parseurXML,
                        "fonctionBaliseOuvrante",
                        "fonctionBaliseFermante");

// Déclaration de la fonction de traitement
// du texte entre les balises
xml_set_character_data_handler($parseurXML, "fonctionTexte");

// Ouverture du fichier
( $fp = fopen($fichier, "r") ) or
    die("Impossible d'ouvrir le fichier XML");

// Lecture du fichier ligne par ligne
while ( $ligneXML = fgets($fp, 1024) ) {
    xml_parse($parseurXML, $ligneXML, feof($fp) ) or
        die("Erreur XML");
}

// Libération des ressources
xml_parser_free($parseurXML);
fclose($fp);
?>
```

Comme vous pouvez le constater, le résultat obtenu est :

```
Noir Désir : des Visages des Figures
- L'enfant roi 6:03
```

```
- Le grand incendie 4:37
- Le vent nous portera 4:48
Placebo : Without You I'm Nothing
- Pure Morning 4:15
- Without You I'm Nothing 4:30
- Every You, Every Me 5:00
```

Mais, surtout, à la lecture du code, vous constatez que la principale difficulté (notamment pour ceux qui n'ont jamais utilisé ce genre de parseur) vient du fait qu'au niveau de la fonction chargée du traitement du texte, nous n'avons, a priori, aucune information sur la balise à laquelle le texte se rapporte. Il convient donc de mémoriser, par nos propres moyens, "l'historique" des balises rencontrées.

Avant de passer à la suite, afin de préparer le terrain pour une utilisation plus complexe, nous vous proposons de légèrement remanier ce script pour en faire un objet.

Mais voilà ! pour utiliser le parseur XML dans un objet, il faut faire appel à la fonction `xml_set_object()`.

xml_set_object()

Permet l'utilisation du parseur XML dans un objet.

Syntaxe	`void xml_set_object(resource $parseur, object &$objet)`
`$parseur`	Référence du parseur telle que retournée par la fonction `xml_parser_create()`.
`$objet`	Référence sur l'objet.

Attention !... suspense... et voilà !

Listing 15.7 : expat_03b.php

```
<?php
class XML_Parseur
{
    var $parseurXML;

    function XML_Parseur()
    {
    }

    function init()
    {
        $this->parseurXML = xml_parser_create();

        // Définit l'objet courant comme étant l'objet
        // qui "héberge" les fonctions de gestion des événements
        xml_set_object($this->parseurXML, &$this);
```

```
    xml_parser_set_option($this->parseurXML, XML_OPTION_CASE_FOLDING,
                          FALSE);
    xml_set_element_handler($this->parseurXML, "fonctionBaliseOuvrante",
                                               "fonctionBaliseFermante");
    xml_set_character_data_handler($this->parseurXML, "fonctionTexte");
}


function parse($fichier)
{
    ( $fp = fopen($fichier, "r") ) or
       die("Impossible d'ouvrir le fichier XML");

    while ( $ligneXML = fgets($fp, 1024) ) {
        xml_parse($this->parseurXML, $ligneXML, feof($fp) ) or
            die("Erreur XML");
    }
    fclose($fp);
}

function free()
{
    xml_parser_free($this->parseurXML);
}

//
// Méthodes privées
//

function fonctionBaliseOuvrante($parseur, $nomBalise, $tableauAttributs)
{
    global $balises;
    $balises[] = $nomBalise;

    switch ($nomBalise) {
        case "cdaudio" :
            echo "<b>";
            echo $tableauAttributs["interprete"];
            echo " : ";
            echo $tableauAttributs["titre"];
            echo "</b><br />";
            break;
    }

    return TRUE;
}

function fonctionBaliseFermante($parseur, $nomBalise)
{
    global $balises;
    array_pop($balises);
```

```
        return TRUE;
    }

    function fonctionTexte($parseur, $texte)
    {
        global $balises;

        $derniereBalise = $balises[count($balises)-1];
        switch ($derniereBalise) {
            case "titre" :
                  echo "- $texte";
                  break;
            case "duree":
                echo " $texte<br />";
                break;
        }

        return TRUE;
    }
}

$fichierXML = "../src_xml/mediatheque_01.xml";

$parseur = new XML_Parseur();
$parseur->init();
$parseur->parse($fichierXML);
$parseur->free();
?>
```

Nous voilà prêts pour aborder la suite.

Gestion des entités externes avec xml_set_external_entity_ref_handler

La fonction `xml_set_external_entity_ref_handler()` vous permet de préciser quelle fonction doit traiter les références aux entités externes. Les entités externes apparaissent dans les documents XML sous la forme `&nomEntite;` et sont déclarées sous la forme `<!ENTITY nomEntite SYSTEM "fichier.xml">`.

xml_set_external_entity_ref_handler()

Déclare auprès du parseur la fonction à appeler lorsque des références à des entités externes sont rencontrées (`&nomEntite;` avec `<!ENTITY nomEntite SYSTEM "fichier.xml">`).

Syntaxe

```
boolean xml_set_external_entity_ref_handler(resource
$parseur, string $fonctionRefEntiteExterne)
```

`$parseur`	Référence du parseur telle que retournée par la fonction `xml_parser_create()`.
`$fonctionRef EntiteExterne`	Nom de la fonction en charge du traitement des références à des entités externes.
`retour`	`TRUE` si l'opération s'est déroulée correctement, `NULL` sinon.

Fonction référence à des entités externes

La fonction (dont le nom sera précisé par `$fonctionRefEntiteExterne`) devant traiter les instructions de traitement aura l'interface suivante :

Syntaxe	`boolean maFonction(resource $parseur, string $nomEntite, string $base, string $idSysteme, $idPublic)`
`$parseur`	Référence du parseur telle que retournée par la fonction `xml_parser_create()`.
`$nomEntite`	Nom de l'entité.
`$base`	Non renseigné.
`$idSysteme`	L'identifiant système de l'entité (en d'autres termes, valeur du champ "`SYSTEM`" ou celui suivant la valeur du champ "`PUBLIC`", c'est-à-dire le document externe pointé).
`$idPublic`	L'identifiant public de l'entité (concrètement, la valeur du champ "`PUBLIC`", c'est-à-dire le document externe pointé servant de référence).
`retour`	`TRUE` si l'opération a été réalisée avec succès, `FALSE` sinon.

L'ensemble des paramètres étant fourni par le parseur, la fonction aura accès à toutes les informations nécessaires au bon traitement des données. Cependant, il est à noter que la présence de cette fonction sous-entend que la lecture et l'inclusion du document externe sont à votre charge.

Listing 15.8 : expat_04.php

```
<?php
class XML_Parseur
{
    var $parseurXML;
    var $fichierXML;    // Fichier sur lequel s'est appliqué
                        // le dernier traitement

    function XML_Parseur()
    {
    }

    function init()
    {
        $this->parseurXML = xml_parser_create();

        // Définit l'objet courant comme étant l'objet
```

```
    // qui "héberge" les fonctions de gestion des événements
    xml_set_object($this->parseurXML, &$this);

    xml_parser_set_option($this->parseurXML, XML_OPTION_CASE_FOLDING,
                          FALSE);
    xml_set_element_handler($this->parseurXML, "fonctionBaliseOuvrante",
                                               "fonctionBaliseFermante");
    xml_set_character_data_handler($this->parseurXML, "fonctionTexte");

   // Déclaration de la fonction de traitement
   // des references aux entités externes
   xml_set_external_entity_ref_handler($this->parseurXML,
                                       "fonctionRefEntiteExterne");

}


function parse($fichier)
{
    // Pour le traitement des references aux entités externes
    // nous avons besoin de mémoriser à quel fichier est appliqué
    // le traitement
    $this->fichierXML = $fichier;

    ( $fp = fopen($fichier, "r") ) or
       die("Impossible d'ouvrir le fichier XML");

    while ( $ligneXML = fgets($fp, 1024) ) {
        xml_parse($this->parseurXML, $ligneXML, feof($fp) ) or
            die("Erreur XML".
                xml_error_string(xml_get_error_code($this->parseurXML)));
    }
    fclose($fp);
}

function free()
{
    xml_parser_free($this->parseurXML);
}

//
// Méthodes privées
//

function fonctionBaliseOuvrante($parseur, $nomBalise, $tableauAttributs)
{
    global $balises;
    $balises[] = $nomBalise;

    switch ($nomBalise) {
        case "cdaudio" :
            echo "<b>";
```

```
            echo $tableauAttributs["interprete"];
            echo " : ";
            echo $tableauAttributs["titre"];
            echo "</b><br />";
            break;
    }

    return TRUE;
}

function fonctionBaliseFermante($parseur, $nomBalise)
{
    global $balises;
    array_pop($balises);

    return TRUE;
}

function fonctionTexte($parseur, $texte)
{
    global $balises;

    $derniereBalise = $balises[count($balises)-1];
    switch ($dernieBalise) {
        case "titre" :
              echo "- $texte";
              break;
        case "duree":
            echo " $texte<br />";
            break;
    }

    return TRUE;
}

// Exemple de fonction gérant
// les références aux entités externes
function fonctionRefEntiteExterne($parseur,
                                  $nomEntite,
                                  $base,
                                  $idSysteme,
                                  $idPublic)
{
    // Il est ici, fait référence à
    // $idSysteme (et eventuellement $idPublic)
    // nous ne nous interesserons ici
    // qu'a $idSysteme.

    // Attention: Petite subtilité
    // le chemin du fichier externe est relatif
    // au fichier XML et non au script PHP
    // (REM: Ici, on suppose que le chemin est relatif
```

```
            // et non absolu)

            $fichierExterne = dirname($this->fichierXML)."/".$idSysteme;

            // Nous instancions un nouveau parseur
            // (cela pourrait par exemple permettre de gérer
            // des fichiers externes possédant un encodage différent)

            $tmpParseur = new XML_Parseur();
            $tmpParseur->parse($fichierExterne);
            $tmpParseur->free();

            // Ne pas oublier de retourner TRUE en cas
            // de succès des opérations.
            return TRUE;
        }
}

$fichierXML = "../src_xml/mediatheque_01b.xml";

$parseur = new XML_Parseur();
$parseur->init();
$parseur->parse($fichierXML);
$parseur->free();
?>
```

Appliqué au fichier *mediatheque_01b.xml* cela donne le sobre résultat suivant (à vous de l'enrichir) :

Noir Désir : des Visages des Figures
- L'enfant roi 6:03
- Le grand incendie 4:37
- Le vent nous portera 4:48

Placebo : Without You I'm Nothing
- Pure Morning 4:15
- Without You I'm Nothing 4:30
- Every You, Every Me 5:00

Radiohead : OK Computer
- Karma Police 4:23

Muse : Showbiz
- Feeling Good Non communiqué

Figure 15.2 :
Résultat de transformation expat

Référence à une entité interne
Les références internes sont, quant à elles, automatiquement remplacées par leurs valeurs (comme le prouve l'exemple précédent).

Gestion des instructions de traitement <? ... ?> avec xml_set_processing_instruction_handler

La fonction `xml_set_processing_instruction_handler()` vous permet de préciser quelle fonction doit traiter les instructions de traitement précisées entre `<?` et `?>`. À noter que cela ne concerne pas les instructions de traitement XML (`<?xml ?>`).

xml_set_processing_instruction_handler()

Déclare auprès du parseur la fonction à appeler lorsque des instructions de traitement (entre <? et ?>) sont rencontrées.

Syntaxe	`boolean xml_set_processing_instruction_handler(resource $parseur, string $fonctionInstructionTraitement)`
`$parseur`	Référence du parseur telle que retournée par la fonction `xml_parser_create()`.
`$fonctionInstructionTraitement`	Nom de la fonction en charge du traitement des instructions de traitement.
`retour`	`TRUE` si l'opération s'est déroulée correctement, `NULL` sinon.

Fonction instruction de traitement

La fonction (dont le nom sera précisé par `$fonctionInstructionTraitement`) devant traiter les instructions de traitement aura l'interface suivante :

Syntaxe	`[boolean] maFonction(resource $parseur, string $cible, string $code)`
`$parseur`	Référence du parseur telle que retournée par la fonction `xml_parser_create()`.
`$cible`	Indique à qui est destinée l'instruction (il s'agit de l'identifiant que l'on trouve juste après "<?". Ce peut être par exemple "`php`").
`$code`	Le code ou tout au moins les instructions.
`retour`	`TRUE` en cas de succès, `FALSE` sinon. (Ceci n'est pas requis ni pris en compte à ce jour par le parseur, mais c'est conseillé puisque cet état de fait peut évoluer).

L'ensemble des paramètres étant fourni par le parseur, la fonction aura accès à toutes les informations nécessaires au bon traitement des données.

Gestion des déclarations de notation <!NOTATION ... > avec xml_set_notation_decl_handler

xml_set_notation_decl_handler()

Déclare auprès du parseur la fonction à appeler lorsque des déclarations de notation (<!NOTATION ...>) sont rencontrées.

Syntaxe	`boolean xml_set_notation_decl_handler(resource $parseur, string $fonctionDeclarationNotation)`
`$parseur`	Référence du parseur telle que retournée par la fonction `xml_parser_create()`.

$fonctionDeclarationNotation	Nom de la fonction en charge du traitement des déclarations de notation.
retour	TRUE si l'opération s'est déroulée correctement, NULL sinon.

Fonction déclaration de notation

La fonction (dont le nom sera précisé par $fonctionDeclarationNotation) devant traiter les instructions de traitement aura l'interface suivante :

Syntaxe	[boolean] maFonction(resource $parseur, string $nomNotation, string $base, string $idSysteme, string $idPublic)
$parseur	Référence du parseur telle que retournée par la fonction xml_parser_create().
$nomNotation	Nom de la notation.
$base	Non renseigné.
$idSysteme	L'identifiant système de l'entité (en d'autres termes la valeur du champ "SYSTEM" ou celui suivant la valeur du champ "PUBLIC". C'est-à-dire la déclaration de type ou l'application de traitement pointée).
$idPublic	L'identifiant public de l'entité (concrètement, la valeur du champ "PUBLIC". C'est-à-dire la déclaration de type ou l'application de traitement pointée servant de référence).
retour	TRUE en cas de succès, FALSE sinon. (Ceci n'est pas requis ni pris en compte à ce jour par le parseur, mais c'est conseillé puisque cet état de fait peut évoluer).

Gestion des déclarations d'entités non analysables <!ENTITY ...NDATA... > avec xml_set_unparsed_entity_decl_handler

xml_set_unparsed_entity_decl_handler()

Déclare auprès du parseur la fonction à appeler lorsque des déclarations d'entités non analysables (<!ENTITY ... NDATA ...>) sont rencontrées.

Syntaxe	boolean xml_set_unparsed_entity_decl_handler(resource $parseur, string $fonctionDeclarationEntiteNonAnalysable)
$parseur	Référence du parseur telle que retournée par la fonction xml_parser_create().
$fonctionDeclarationEntiteNonAnalysable	Nom de la fonction en charge du traitement des déclarations d'entités non analysables.
retour	TRUE si l'opération s'est déroulée correctement, NULL sinon.

Fonction déclaration d'entité non analysable

La fonction (dont le nom sera précisé par `$fonctionDeclarationEntiteNonAnalysable`) devant traiter les instructions de traitement aura l'interface suivante :

Syntaxe	`[boolean] maFonction(resource $parseur, string $nomEntite, string $base, string $idSysteme, string $idPublic, string $nomNotation)`
`$parseur`	Référence du parseur telle que retournée par la fonction `xml_parser_create()`.
`$nomEntite`	Nom de l'entité.
`$base`	Non renseigné.
`$idSysteme`	L'identifiant système de l'entité (en d'autres termes la valeur du champ "`SYSTEM`" ou celui suivant la valeur du champ "`PUBLIC`". C'est-à-dire la déclaration de type ou l'application de traitement pointée).
`$idPublic`	L'identifiant public de l'entité (concrètement, la valeur du champ "`PUBLIC`". C'est-à-dire la déclaration de type ou l'application de traitement pointée servant de référence).
`$nomNotation`	Nom de la notation.
`retour`	`TRUE` en cas de succès, `FALSE` sinon. (Ceci n'est pas requis ni pris en compte à ce jour par le parseur, mais c'est conseillé puisque cet état de fait peut évoluer).

Gestion par défaut avec xml_set_default_handler

La fonction `xml_set_default_handler()` vous permet de préciser quelle fonction doit être appelée par défaut (cela concerne aussi bien les événements pour lesquels il n'existe pas de fonction `xml_..._handler` associée que les événements ayant leur propre fonction `xml_..._handler`, mais pour lesquels vous n'avez pas défini de fonction de traitement).

xml_set_default_handler()

Déclare auprès du parseur la fonction à appeler pour le traitement par défaut.

Syntaxe	`boolean xml_set_default_handler(resource $parseur, string $fonctionParDefaut)`
`$parseur`	Référence du parseur telle que retournée par la fonction `xml_parser_create()`.
`$fonctionParDefaut`	Nom de la fonction en charge du traitement par défaut.

Fonction par défaut

La fonction (dont le nom sera précisé par `$fonctionParDefaut`) devant traiter les instructions de traitement aura l'interface suivante :

Syntaxe	`void maFonction(resource $parseur, string $chaine)`
`$parseur`	Référence du parseur telle que retournée par la fonction `xml_parser_create()`.
`$chaine`	Chaîne de caractères contenant la chaîne non reconnue comme événement prédéfini.

L'ensemble des paramètres étant fourni par le parseur, la fonction aura accès à toutes les informations nécessaires au bon traitement des informations.

Gestion des erreurs

Jusque-là, afin de garder une certaine continuité dans le raisonnement, nous avons passé sous silence les fonctions de gestion des erreurs. Pourtant, elles sont souvent indispensables pour déterminer les causes des problèmes d'analyse des documents.

À tout moment, il est possible de récupérer le dernier code d'erreur rencontré par un parseur lors de l'analyse. Pour cela, il y a la fonction `xml_get_error_code()`.

xml_get_error_code()

Retourne le dernier code d'erreur rencontré par le parseur.

Syntaxe	`int xml_get_error_code(resource $parseur)`
`$parseur`	Référence du parseur telle que retournée par la fonction `xml_parser_create()`.
`retour`	Code d'erreur pouvant prendre les valeurs : `XML_ERROR_NONE` : pas d'erreur. `XML_ERROR_NO_MEMORY` : plus de mémoire disponible. `XML_ERROR_SYNTAX` : erreur de syntaxe. `XML_ERROR_NO_ELEMENTS` : aucune balise n'a été rencontrée. `XML_ERROR_INVALID_TOKEN` : document non "bien formé". `XML_ERROR_UNCLOSED_TOKEN`. `XML_ERROR_PARTIAL_CHAR`. `XML_ERROR_TAG_MISMATCH` : présence de balises non fermées ou se chevauchant (ex. : `<a><b></a></b>`). `XML_ERROR_DUPLICATE_ATTRIBUTE` : présence multiple du même attribut dans une même balise. `XML_ERROR_JUNK_AFTER_DOC_ELEMENT` : présence de balises après la balise racine fermante. `XML_ERROR_PARAM_ENTITY_REF`. `XML_ERROR_UNDEFINED_ENTITY` : entité non définie. `XML_ERROR_RECURSIVE_ENTITY_REF` : référence d'entité récursive. `XML_ERROR_ASYNC_ENTITY`. `XML_ERROR_BAD_CHAR_REF`. `XML_ERROR_BINARY_ENTITY_REF` : référence à une entité binaire (entité non analysable non texte).

XML_ERROR_ATTRIBUTE_EXTERNAL_ENTITY_REF.
XML_ERROR_MISPLACED_XML_PI.
XML_ERROR_UNKNOWN_ENCODING : encodage inconnu.
XML_ERROR_INCORRECT_ENCODING : encodage ne correspondant pas au document.
XML_ERROR_UNCLOSED_CDATA_SECTION.
XML_ERROR_EXTERNAL_ENTITY_HANDLING : la fonction de gestion des entités externes n'a pas retourné TRUE (laissant supposer qu'une erreur est intervenue), ou NULL si le parseur n'existe pas.

Il est également possible d'avoir l'information sous forme de texte (en anglais) via xml_error_string().

xml_error_string()

Retourne le message d'erreur (en anglais) correspondant au code d'erreur fourni.

Syntaxe	`string xml_error_string(int $codeErreur)`
`$codeErreur`	Code d'erreur tel que retourné par `xml_error_code()`.
`retour`	Le message d'erreur, ou `NULL` si le code d'erreur n'existe pas.

Pour aider à l'analyse du problème, il est souvent également nécessaire de connaître la position estimée de l'erreur. Heureusement, il est à tout moment (donc pas uniquement en cas d'erreur) possible de connaître la position du parseur. Pour cela, vous disposez de `xml_get_current_line_number()` et `xml_get_current_column_number()`.

xml_get_current_line_number()

Retourne le numéro de la ligne en cours d'analyse par le parseur.

Syntaxe	`int xml_get_current_line_number(resource $parseur)`
`$parseur`	Référence du parseur telle que retournée par la fonction `xml_parser_create()`.
`retour`	Numéro de ligne, ou `NULL` si le parseur n'existe pas.

xml_get_current_column_number()

Retourne le numéro de la colonne en cours d'analyse par le parseur.

Syntaxe	`int xml_get_current_column_number(resource $parseur)`

`$parseur`	Référence du parseur telle que retournée par la fonction `xml_parser_create()`.
`retour`	Numéro de colonne, ou `NULL` si le parseur n'existe pas.

Et, pour ceux que cela intéresse, il est également possible d'avoir la position en octets depuis le début du document avec `xml_get_current_byte_index()`.

Le parseur DOM

Si libXML est installé, alors le parseur DOM est probablement activé. Vous pouvez le vérifier en jetant un œil aux informations affichées par `phpinfo()`.

dom	
DOM/XML	enabled
DOM/XML API Version	20031129
libxml Version	2.5.11
HTML Support	enabled
XPath Support	enabled
XPointer Support	enabled
Schema Support	enabled
RelaxNG Support	enabled

Figure 15.3 : *phpinfo() DOM*

Parfois, il est préférable d'accéder aux parties d'un fichier XML dans un ordre différent de celui imposé par SAX. L'avantage de l'analyse par DOM est que l'on peut accéder à tout élément de l'arbre très facilement et même faire des retours en arrière. L'inconvénient (et c'est probablement pourquoi SAX et DOM coexistent) c'est que l'arbre entier est mis en mémoire, ainsi pour un gros fichier XML, DOM n'est pas adapté.

La bibliothèque DOM étant imposante, nous nous limiterons exceptionnellement aux fonctionnalités les plus utilisées.

Constantes

Voici l'ensemble des constantes définies:

Tableau 15.1 : Constantes XML

Nom de variable	Définition
`XML_ELEMENT_NODE`	Nœud de type Elément.
`XML_ATTRIBUTE_NODE`	Nœud de type Attribut.
`XML_TEXT_NODE`	Nœud de type Text.
`XML_CDATA_SECTION_NODE`	Nœud de type section CData (non traitée)
`XML_ENTITY_REF_NODE`	
`XML_ENTITY_NODE`	Nœud de type Entité (comme é …)
`XML_PI_NODE`	
`XML_COMMENT_NODE`	Nœud de type Commentaire

Nom de variable	Définition
XML_DOCUMENT_NODE	Noeud de type Document
XML_DOCUMENT_TYPE_NODE	
XML_DOCUMENT_FRAG_NODE	
XML_NOTATION_NODE	
XML_HTML_DOCUMENT_NODE	Nœud d'un document HTML
XML_DTD_NODE	Nœud d'une DTD
XML_ELEMENT_DECL_NODE	
XML_ATTRIBUTE_DECL_NODE	
XML_ENTITY_DECL_NODE	
XML_NAMESPACE_DECL_NODE	
XML_ATTRIBUTE_CDATA	
XML_ATTRIBUTE_ID	
XML_ATTRIBUTE_IDREF	
XML_ATTRIBUTE_IDREFS	
XML_ATTRIBUTE_ENTITY	
XML_ATTRIBUTE_NMTOKEN	
XML_ATTRIBUTE_NMTOKENS	
XML_ATTRIBUTE_ENUMERATION	
XML_ATTRIBUTE_NOTATION	

Tableau 15.2 : Constantes DOM

Nom de variable	Définition
DOM_INDEX_SIZE_ERR	L'index ou la taille est négative ou plus grande que la valeur acceptée.
DOMSTRING_SIZE_ERR	
DOM_HIERARCHY_REQUEST_ERR	Un nœud est inséré un endroit où il ne devrait pas.
DOM_WRONG_DOCUMENT_ERR	Un nœud est utilisé dans un autre document que celui où il a été créé.
DOM_INVALID_CHARACTER_ERR	Un caractère invalide a été inséré.
DOM_NO_DATA_ALLOWED_ERR	Des données sont spécifiées pour un nœud qui n'en accepte pas.
DOM_NO_MODIFICATION_ALLOWED_ERR	Il y a eu tentative de modification à un endroit interdit.

Nom de variable	Définition
DOM_NOT_FOUND_ERR	Il y a eu tentative de référencer un nœud dans un contexte où il n'existe pas.
DOM_NOT_SUPPORTED_ERR	Le type d'objet demandé n'est pas supporté.
DOM_INUSE_ATTRIBUTE_ERR	Il y a eu tentative d'ajouter un attribut qui est déjà utilisé à un autre endroit.
DOM_INVALID_STATE_ERR	
DOM_SYNTAX_ERR	
DOM_INVALID_MODIFICATION_ERR	
DOM_NAMESPACE_ERR	
DOM_INVALID_ACCESS_ERR	
DOM_VALIDATION_ERR	

Les différentes classes principales

Voici quelques unes des classes les plus utilisées avec leurs attributs et leurs fonctions brièvement présentées. Les fonctions les plus importantes sont par ailleurs décrites en détails à la suite de ces tableaux succincts.

Tableau 15.3 : Classe DOMDocument

Attribut	Type	Définition
doctype	String	
implementation	DOMImplemenation	
documentElement	DOMElement	Elément racine de l'arbre.
actualEncoding	String	Encodage utilisé.
encoding	String	
standalone	Boolean	
version	String	Version XML utilisée.
strictErrorChecking	Boolean	
documentURI	String	
config	String	
formatOutput	String	
validateOnParse	Boolean	
resolveExternals	Boolean	
preserveWhiteSpace	Boolean	
substituteEntities	Boolean	

Attribut	Type	Définition
`createAttribute(String $nom)`	DOMAttr	Crée un nouvel attribut.
`createAttributeNS(String $espaceNom, String $nom)`	DOMAttr	Crée un nouvel attribut avec espace de nom.
`createCDATASection(String $donnees)`	DOMCDATASection	Crée une nouvelle section CDATA.
`createComment(String $commentaire)`	DOMComment	Crée un nouveau commentaire.
`createDocumentFragment()`		Crée un nouveau fragment.
`createElement(String $nom [, String $valeur])`	DOMElement	Crée un nouvel élément.
`createElementNS(String $espaceNom, String $nom)`	DOMElement	Crée un nouvel élément avec espace de noms.
`createEntityReference(String $nom)`	DOMEntityReference	Crée une nouvelle entité.
`createProcessingInstruction (String $cible [, String $donnees])`	DOMProcessingInstruction	Crée un nœud de type PI.
`createTextNode(String $texte)`	DOMText	Crée un nœud de type texte.
`getElementById(String $idElement)`	DOMElement	Fait une recherche sur l'identifiant de l'élément. Selon le standard DOM cette fonction requière qu'un attribut ID de type ID soit défini dans la DTD.
`getElementByTagName(String $nom)`	DOMNodeList	Retourne les nœuds ayant $nom comme nom.
`getElementByTagNameNS(String $espaceNom, String $nom)`	DOMNodeList	Idem mais avec espace de noms.
`importNode(DOMNode $noeudImporte [, boolean $copieProfonde])`	DOMNode	Importe un nœud dans le document courrant
`load(String $nomFichier)`	DOMDocument	Charge et crée l'arbre DOM à partir du nom de fichier d'un document XML.
`loadHTML(String $HTML)`	DOMDocument	Charge et crée l'arbre DOM à partir de la chaîne de caractères $HTML.

Attribut	Type	Définition
`loadHTMLFile(String $NomFichierHTML)`	DOMDocument	Charge et crée l'arbre DOM à partir du nom de fichier d'un document HTML.
`loadXML(String $XML)`	DOMDocument	Charge et crée l'arbre DOM à partir de la chaîne de caractères $XML.
`normalize()`	void	Normalise le document.
`relaxNGValidate(String $nomFichier)`	boolean	Vérifie la validité d'un document selon le fichier relaxNG passé en paramètre.
`relaxNGValidateSource(String $relaxNG)`	boolean	Vérifie la validité d'un document selon relaxNG passé en paramètre.
`save(String $nomFichier)`	integer	Crée et stocke dans un fichier le code XML correspondant au document DOM.
`saveHTML()`	String	Crée le HTML correspondant au document DOM.
`saveHTMLFile(String $nomFichier)`	String	Crée et stocke dans un fichier le code HTML correspondant au document DOM.
`saveXML([DOMNode $noeud])`	String	Crée le XML correspondant au document DOM.
`schemaValidate(String $nomFichierSchema)`	boolean	Vérifie la validité d'un document selon le fichier Schema passé en paramètre.
`schemaValidateSource(String $schema)`	boolean	Vérifie la validité d'un document selon un Schema passé en paramètre.
`validate()`	boolean	Vérifie si un le document est valide selon sa DTD.
`xinclude()`	int	Remplace les xinclude dans un document DOM.

Tableau 15.4 : Classe DOMNode

Attribut	Type	Définition
`nodeName`	String	Nom du nœud.
`nodeValue`	String	Valeur du nœud (contenu).
`nodeType`	Integer	Code type du Nœud.
`parentNode`	DOMElement	Nœud parent.
`childNodes`	DOMNodeList	Nœuds enfants.
`firstChild`	DOMNode	Premier enfant.
`lastChild`	DOMNode	Dernier enfant.
`previousSibling`	DOMNode	Nœud précédant de même hiérarchie.
`nextSibling`	DOMNode	Enfant suivant de même hiérarchie.
`attributes`	DOMNamedNodeMap	Attributs du nœud.
`ownerDocument`	DOMDocument	Document auquel appartient le nœud.
`namespaceURI`	String	Espace de noms (nom complet).
`prefix`	String	Espace de noms (nom réduit préfix du nœud) si utilisé.
`localName`	String	Nom du nœud sans le préfix d'espace de nom si le préfix est utilisé.
`baseURI`	String	
`textContent`	String	Contenu du nœud.
`appendChild(DOMNode $noeud)`	DOMNode	Ajoute un fils à la suite des autres fils du même élément.
`cloneNode([boolean $enProfondeur])`	DOMNode	Fait une copie du nœud courant.
`hasAttributes()`	boolean	Vérifie si le nœud a des attributs.
`hasChildNodes()`	DOMNode	Vérifie si le nœud a des enfants.
`insertBefore(DOMNode $noeudAInserer[, DOMNode $noeudReference])`	DOMNode	Insert le nœud $noeudAInserer juste avant $noeudReference.
`isSameNode($DOMNode $noeudAComparer)`	boolean	Renvoi TRUE si les deux nœuds sont les mêmes. (pas au niveau du contenu)

Attribut	Type	Définition
`isSupported(string $option, String $version)`	boolean	Renvoi TRUE si l'option est disponible dans la version précisée.
`lookupNamespaceURI(String $prefix)`	String	Retourne l'uri du préfix passé en paramètre.
`lookupPrefix(String $espaceNoms)`	String	Retourne le prefix de l'uri passé en paramètre.
`normalize()`	void	Normalise le nœud.
`removeChild(DOMNode $fils)`	DOMNode	Retire le fils passé en paramètre du nœud courant.
`replaceChild(DOMNode $ancienNoeud, DOMNode $nouveauNoeud)`	DOMNode	Remplace l'ancien nœud par le nouveau nœud.

Tableau 15.5 : Classe DOMElement héritante de DOMNode

Attribut	Type	Définition
`tagName`	String	Nom de la balise
`schemaTypeInfo`	String	
`getAttribute(String $nom)`	String	Retourne la valeur d'un attribut.
`getAttributeNode(String $nom)`	DOMAttr	Retourne le nœud d'un attribut.
`getAttributeNodeNS(String $espaceNoms, String $nom)`	DOMAttr	Retourne le nœud d'un attribut dans un espace de noms.
`getAttributeNS(String $espaceNoms, String $nom)`	String	Retourne la valeur d'un attribut en spécifiant l'espace de noms.
`getElementByTagName(String $nom)`	DOMNodeList	Retourne les éléments de nom $nom.
`getElementByTagNameNS(String $espaceNom, String $nom)`	DOMNodeList	Idem mais avec espace de noms.
`hasAttribute(String $nom)`	boolean	Retourne TRUE si l'attribut de nom $nom existe.
`hasAttributeNS(String $espaceNoms, String $nom)`	boolean	Retourne TRUE si l'attribut de nom $nom existe dans l'espace de nom spécifié.
`removeAttribute(String $nom)`	boolean	Retire l'attribut de nom $nom.

Attribut	Type	Définition
`removeAttributeNode(DOMNode $nom)`	boolean	Retire l'attribut de nom $nom.
`removeAttributeNS(String $espaceNoms, String $nom)`	boolean	Retire l'attribut de nom $nom dans l'espace de noms.
`setAttribute(String $nom, String $valeur)`	boolean	Crée ou met à jour l'attribut $nom avec la valeur $valeur.
`setAttributeNode(DOMAttr $attribut)`	boolean	Ajoute l'attribut $attribut.
`setAttributeNodeNS(DOMAttr $attribut)`	boolean	Ajoute l'attribut $attribut.
`setAttributeNS(String $espaceNoms, String $nom, String $valeur)`	void	Crée ou met à jour l'attribut $nom avec la valeur $valeur dans l'espace de noms spécifié.

Tableau 15.6 : Classe DOMAttr

Attribut	Type	Définition
`name`	String	Nom de l'attribut.
`specified`	Boolean	
`value`	String	Valeur de l'attribut.
`ownerElement`	DOMElement	Elément auquel est rattaché l'attribut.
`schemaTypeInfo`	String	
`isId()`	boolean	Renvoi TRUE si l'attribut est un identifiant selon les spécifications DOM. (Défini en tant que tel dans la DTD)

Tableau 15.7 : Classe DOMText héritante de DOMNode

Attribut	Type	Définition
`wholeText`	String	
`isWhitespaceInElementContent()`	boolean	Renvoi TRUE si les espaces font parti du contenu.

Attribut	Type	Définition
`splitText(integer $index)`	DOMText	"Casse" le nœud en deux à l'indice spécifié le nœud sur lequel est appelé cette fonction contient alors la première partie tandis que la seconde partie est retournée.

Créer ou modifier un arbre DOM

Pour créer un arbre il faut d'abord créer un objet de type `DOMDocument`. (par exemple: `$document = new DOMDocument()`)

DOMDocument()

Constructeur d'objet de type DOMDocument.

Syntaxe	`DOMDocument DOMDocument([String $versionXML])`
`$versionXML`	Version XML du document.
`retour`	Objet de type DOMDocument.

A partir d'un document existant, il va falloir le charger en mémoire, pour cela on utilise `DOMDocument->load()`.

DOMDocument->load()

Permet de charger un fichier XML en mémoire. Cette fonction peut aussi s'appeler statiquement.

Syntaxe	`DOMDocument load(String $nomFichier)`
`$nomFichier`	Nom du fichier à charger.
`retour`	Un nouveau DOMDocument si la fonction a été appelée de manière statique.

Pour charger une chaîne de caractère XML, autrement que dans un fichier, il faut faire appel à `DOMDocument->loadxml()`.

DOMDocument->loadXml()

Permet de charger une chaîne de caractères XML en mémoire. Cette fonction peut aussi s'appeler statiquement.

Syntaxe	`DOMDocument loadXml(String $chaineXML)`
`$chaineXML`	Chaîne de caractères à charger.
`retour`	Un nouveau DOMDocument si la fonction a été appelée de manière statique.

Un fichier HTML est très souvent mal formé, par exemple on y trouve des balises `<br>` (si ce n'est pas de l'XHTML) pour charger du HTML on utilise donc `DOMDocument->loadhtmlfile()` et `DOMDocument->loadhtml()`.

DOMDocument->loadHTMLFile()

Permet de charger un fichier HTML en mémoire. Cette fonction peut aussi s'appeler statiquement.

Syntaxe	`DOMDocument loadHTMLFile(String $nomFichierHTML)`
`$nomFichierHTML`	Nom du fichier HTML à charger.
`retour`	Un nouveau DOMDocument si la fonction a été appelée de manière statique.

DOMDocument->loadHTML()

Permet de charger une chaîne de caractères HTML en mémoire. Cette fonction peut aussi s'appeler statiquement.

Syntaxe	`DOMDocument loadHTML(String $chaineHTML)`
`$chaineHTML`	Chaîne de caractères HTML à charger.
`retour`	Un nouveau DOMDocument si la fonction a été appelée de manière statique.

Les quatre fonctions suivantes permettent la création du XML (ou HTML) à partir de l'arbre DOM:

DOMDocument->save()

Permet de sauvegarder l'arbre DOM dans un fichier.

Syntaxe	`int save(String $nomFichier)`
`$nomFichier`	Nom du fichier où enregistrer l'arbre DOM.
`retour`	La taille en octets du fichier.

DOMDocument->saveXML()

Permet de créer la chaîne XML correspondante à l'arbre DOM en mémoire.

Syntaxe	`String saveXML([DOMNode $noeud])`
`$noeud`	Nœud parent du XML à extraire si l'on ne veut pas l'arbre entier.
`retour`	Chaîne de caractères XML.

DOMDocument->saveHTMLFile()

Permet de sauvegarder l'arbre DOM dans un fichier.

Syntaxe	`String saveHTMLFile(String $nomFichier)`
`$nomFichier`	Nom du fichier où enregistrer l'arbre DOM.
`retour`	Chaîne de caractères HTML.

DOMDocument->saveHTML()

Permet de créer la chaîne HTML correspondante à l'arbre DOM en mémoire.

Syntaxe	`String saveHTML()`
`retour`	Chaîne de caractères HTML.

A partir d'un tel document, il est possible de créer des instances d'objets constituant l'arbre. En voici les principales fonctions:

DOMDocument->createAttribute()

Permet de créer un attribut. Il faudra attacher cet attribut à un nœud en utilisant `DOMNode->appendchild()`.

Syntaxe	`DOMAttr createAttribute(String $nomAttribut)`
`$nomAttribut`	Nom de l'attribut.
`retour`	Un nouveau objet de type DOMAttr. FALSE en cas d'erreur

DOMDocument->createAttributeNS()

Permet de créer un attribut avec espace de noms. Il faudra attacher cet attribut à un nœud en utilisant `DOMNode->appendchild()`.

Syntaxe	`DOMAttr createAttributeNS(String $espaceNoms, String $nomAttribut)`
`$espaceNoms`	Espace de noms
`$nomAttribut`	Nom de l'attribut.
`retour`	Un nouveau objet de type DOMAttr. FALSE en cas d'erreur

DOMDocument->createCDATASection ()

Permet de créer une section CDATA, ce type de section peut servir pour inclure des données XML encodées (Base64 est souvent utilisé) ou simplement du texte qui pourrait nuire au fichier XML (car contenant des balises par exemple). Il faudra attacher cet attribut à un nœud en utilisant `DOMNode->appendchild()`.

Syntaxe	`DOMCDataSection createCDATASection(String $donnees)`
`$donnees`	Données à mettre dans la section.
`retour`	Un nouveau objet de type DOMCDataSection. FALSE en cas d'erreur

DOMDocument->createComment()

Permet de créer une balise de commentaires (entre <!-- et ->. Il faudra attacher cet attribut à un nœud en utilisant `DOMNode->appendchild()`.

Syntaxe	`DOMComment createComment(String $commentaires)`

`$commentaires`	Commentaires à ajouter.
`retour`	Un nouveau objet de type DOMComment. FALSE en cas d'erreur

DOMDocument->createElement()

Permet de créer un nouvel élément. Il faudra attacher cet attribut à un nœud en utilisant `DOMNode->appendchild()`.

Syntaxe	`DOMElement createElement(String $nomElement [, String $valeur])`
`$nomElement`	Nom de la balise.
`$valeur`	Valeur de la balise
`retour`	Un nouveau objet de type DOMElement. FALSE en cas d'erreur

DOMDocument->createElementNS()

Permet de créer un nouvel élément avec espace de noms. Il faudra attacher cet attribut à un nœud en utilisant `DOMNode->appendchild()`.

Syntaxe	`DOMElement createElementNS(String $espaceNoms, String $nomElement)`
`$espaceNoms`	Espace de noms
`$nomElement`	Nom de la balise.
`retour`	Un nouveau objet de type DOMElement. FALSE en cas d'erreur

DOMDocument->createTextNode()

Permet de créer un nouvel nœud texte. Il faudra attacher cet attribut à un nœud en utilisant `DOMNode->appendchild()`.

Syntaxe	`DOMText createTextNode(String $texte)`
`$texte`	Texte du nœud.
`retour`	Un nouveau objet de type DOMText. FALSE en cas d'erreur

Voici un exemple d'arbre DOM construit puis affiché:

Listing 15.9 : dom_exemple_01.php

```
<?php
  $doc = new DOMDocument();
  $html = $doc->createElement("HTML");
  $html = $doc->appendChild($html);
  $head = $doc->createElement("HEAD");
  $head = $html->appendChild($head);
  $title = $doc->createElement("TITLE");
  $title = $head->appendChild($title);
  $titre = $doc->createTextNode("Titre du document HTML");
  $titre = $title->appendChild($titre);
  $body = $doc->createElement("BODY");
  $body = $doc->appendChild($body);
  $comment = $doc->createComment("Comme en terre");
  $comment = $body->appendChild($comment);
  echo $doc->saveHTML();
?>
```

Dont le résultat est:

```
<HTML><HEAD><TITLE>Titre du document HTML</TITLE></HEAD></HTML>
<BODY><!--Comme en terre--></BODY>
```

Le script est extrêmement simple à comprendre même si nous n'avons pas encore vu la fonction `appendChild()` qui ne fait qu'ajouter un nœud à un autre, ce nœud devenant donc le fils du second. Les nœuds du document HTML sont créés un à un comme tout document HTML. La première balise est donc `<HTML>` à laquelle on ajoute les balises `<HEAD>` puis `<BODY>`, à la balise `<HEAD>` on ajoute la balise `<TITLE>` et à cette dernière on ajoute un nœud de type texte pour définir son contenu. Enfin on ajoute un commentaire à la balise `<BODY>`.


REMARQUE

Rien à fermer
Pour les personnes qui ne sont pas habituées à manipuler des arbres DOM, vous remarquerez qu'il n'y a pas à "fermer" quoi que ce soit comme c'est le cas pour les balises. Ici, tout fonctionne selon le principe d'imbrication d'éléments.

Nous venons de voir les principales fonctions relatives à l'objet `DOMDocument`, voici celles de `DOMNode` en débutant par celle certainement la plus usitée: `DOMNode->appendChild()`.

DOMNode->appendChild()

Ajoute un nœud enfant à un nœud existant.

Syntaxe	`DOMNode appendChild(DOMNode $noeudEnfant)`
`$noeudEnfant`	Nœud à ajouter au parent.
`retour`	L'objet parent modifié.

Pour, au contraire, retirer un nœud, on utilise `DOMNode->removeChild()` et pour remplacer par un autre: `DOMNode->replaceChild()`.

DOMNode->removeChild()

Retire un nœud de son parent.

Syntaxe	`DOMNode removeChild(DOMNode $noeudEnfant)`
`$noeudEnfant`	Nœud à retirer du parent.
`retour`	L'ancien fils ou une erreur de type DOMException.

DOMNode->replaceChild()

Remplace un nœud fils par un autre.

Syntaxe	`DOMNode replaceChild(DOMNode $ancienNoeud, DOMNode $nouveauNoeud)`
`$ancienNoeud`	Nœud à remplacer.
`$nouveauNoeud`	Nœud remplaçant.
`retour`	L'ancien fils ou une erreur de type DOMException.

Cela peut servir de dupliquer un nœud pour cela il suffit d'utiliser `DOMNode->cloneNode()`.

DOMNode->cloneNode()

Ajoute un nœud enfant à un nœud existant.

Syntaxe	`DOMNode cloneNode([boolean $copieProfondeur])`
`$copieProfondeur`	Indique si le nœud doit être copié avec ses enfants. (Non par défaut)
`retour`	Le nœud copié.

Récupérer des données d'un arbre existant

Récupérer des données d'un arbre DOM est bien plus simple que d'utiliser SAX. La fonction `DOMElement->getElementsByTagName()` permet de retourner tous les nœud ayant le nom spécifié. L'objet retourné est de type `DOMNodeList` qui est un objet très simple avec `length` comme attribut qui retourne le nombre d'élément et `DOMNodeList->item(integer $i)` comme fonction qui permet de retourner le i-ème élément.

DOMElement->getElementsByTagName()

Retourne une liste de nœud dont le nom est la chaîne de caractères passée en paramètre à cette fonction.

Syntaxe	`DOMNodeList getElementsByTagName(String $nom)`
`$nom`	Nom des nœuds à retourner.
`retour`	Une liste de nœuds ayant $nom comme nom.

Afin de récupérer l'attribut d'un élément, il existe la fonction `DOMElement->getAttribute()` détaillée ci-dessous.

DOMElement->getAttribute()

Permet de récupérer la valeur d'un attribut.

Syntaxe :	`String getAttribute(String $nom)`
`$nom`	Nom de l'attribut à retourner
`retour`	Valeur de l'attribut.

Pour récupérer une valeur on peut aussi utiliser son chemin XPath.

DOMXPath->query()

Retourne la liste des nœuds correspondants á l'expression XPath fournie.

Syntaxe :	`DOMNodelist query(String $xpath [, DOMNode $noeudRelatif])`
`$xpath`	Expression Xpath
`$noeudRelatif`	Nœud de référence pour une expression XPath relative.
`retour`	Liste des nœuds

XPath
XPath est très riche et n'est pas présenté ici, mais si vous ne connaissez pas ces expressions utiles pour faire des recherches dans un document, je vous conseille vivement de jeter un œil à:

http://xmlfr.org/w3c/TR/xpath/
pour la documentation de référence traduite en français.
http://www.zvon.org/xxl/XPathTutorial/General_fre/examples.html
pour une aide rapide par l'exemple, ou encore le lien suivant qui met en parallèle requêtes SQL et leur équivalent utilisant XPATH.
http://www.fragbase.com/sql2xpath.php

Il y a bien d'autres fonctions comme présentées dans les tableaux succincts au début de ce chapitre. Et comme un exemple vaut des pages d'explications, en voici un qui montre comment manipuler du XML grâce à DOM.

Exemples

Reprenons l'exemple de la médiathèque du chapitre précédent dont le fichier s'appelait *mediathèque_01.xml*.

Listing 15.10 : dom_exemple_02.php

```
<?php
  $doc = DomDocument::load("mediatheque_01.xml");
  $listeCD = $doc->getElementsByTagName("cdaudio");
  foreach($listeCD as $cdAudio) {
    $interprete = $cdAudio->getAttribute("interprete");
    $listeChansons = $cdAudio->getElementsByTagName("chanson");
    echo "** $interprete\n";
    for ($i=0; $i<$listeChansons->length; $i++) {
      $chanson = $listeChansons->item($i);
      $titre = $chanson->getElementsByTagName("titre")->item(0)->nodeValue;
      echo "  - $titre\n";
    }
  }
?>
```

Dont le résultat attendu serait:

```
** Noir Désir
  - L'enfant roi
  - Le grand incendie
  - Le vent nous portera
** Placebo
  - Pure Morning
  - Without You I'm Nothing
  - Every You, Every Me
```

Quelques explications de cet exemple: tout d'abord on récupère l'objet DOMDocument correspondant au fichier XML stocké sur le disque dur par un appel à la méthode statique `load()`. Ensuite `getElementsByTagName("cdaudio")` permet de récupérer l'ensemble des balises "cdaudio" puis on boucle sur cette liste. Récupérer le nom de l'interprète revient à récupérer la valeur de l'attribut "interprete" de la balise "cdaudio". La liste des chansons du CD

est ensuite récupérée et ici j'ai volontairement changé la façon d'itérer sur les chansons pour utiliser les méthodes de `DOMNodeList`.

Voici maintenant un court exemple de l'utilisation de XPath:

Listing 15.11 : dom_exemple_03.php

```
<?php
  $doc = DomDocument::load("mediatheque_01.xml");
  $xpath = new DOMXpath($doc);
  $liste = $xpath->query("//titre");
  foreach ($liste as $title) {
    echo $title->nodeValue."\n";
  }
?>
```

Dont le résultat attendu serait:

```
L'enfant roi
Le grand incendie
Le vent nous portera
Pure Morning
Without You I'm Nothing
Every You, Every Me
```

Voilà pour l'analyse syntaxique utilisant DOM. Ci-après est décrit en détail la toute nouvelle librairie de PHP5 appelée SimpleXML, elle permet de simplifier l'analyse syntaxique d'un document XML. Jetez-y au moins un œil et vous vous rappellerez pourquoi vous aimez autant le PHP.

Le parseur SimpleXML

Si libXML est installé alors le parseur SimpleXML est probablement activé, vous pouvez le vérifier en jetant un œil aux informations affichées par `phpinfo()`.

SimpleXML

Simplexml support	enabled
Revision	$Revision: 1.139 $
Schema support	enabled

Figure 15.4 : *phpinfo()SimpleXML*

SimpleXML est intégré à PHP depuis la version 5, cette librairie basée sur libxml à pour objectif de simplifier l'analyse syntaxique d'un document XML. Il est ainsi possible en quelques lignes d'accéder à tout élément d'un document XML.

SimpleXML n'est ni plus ni moins qu'un analyseur DOM simplifié.

Exemples

Voici tout d'abord quelques exemples pour voir le fonctionnement de cette bibliothèque. Le document à analyser sera le suivant pour tous les exemples qui suivent et est représentatif de ce que l'on pourrait avoir pour une vidéothèque :

Listing 15.12 : videotheque.xml

```
<?xml version="1.0" standalone="yes" ?>
<videotheque>
  <film imdb="0378194">
    <titre>Kill Bill: Vol. 2</titre>
    <realisateur>Quentin Tarantino</realisateur>
    <genres>
      <genre>action</genre>
      <genre>drama</genre>
      <genre>thriller</genre>
    </genres>
    <acteurs>
      <acteur>
        <nom>Uma Thurman</nom>
        <role>The Bride (Black Mamba)</role>
      </acteur>
      <acteur>
        <nom>David Carradine</nom>
        <role>Bill (Snake Charmer)</role>
      </acteur>
    </acteurs>
  </film>
  <film imdb="0070356">
    <titre>Mais ou est donc passee la septieme compagnie ?</titre>
    <realisateur>Robert Lamoureux</realisateur>
    <genres>
      <genre>war</genre>
      <genre>action</genre>
      <genre>comedy</genre>
    </genres>
    <acteurs>
      <acteur>
        <nom>Jean Lefebvre</nom>
        <role>Pitivier</role>
      </acteur>
      <acteur>
        <nom>Pierre Mondy</nom>
        <role>Sergent Chaudard</role>
      </acteur>
    </acteurs>
  </film>
</videotheque>
```

Le premier exemple montre comment récupérer un élément du document XML, ici le titre du premier film, le titre du second film et le nom du deuxième acteur du second film. Plusieurs

façons d'accéder à ses données sont présentées, en effet, l'indice du tableau n'est pas obligatoire quand on veut accéder au premier élément, c'est particulièrement utile lorsqu'il n'y a qu'un élément comme "titre", "realisateur", "genres"…

Listing 15.13 : parse_videotheque_1.php

```
<?php
$xml = simplexml_load_file("videotheque.xml");
echo $xml->film[0]->titre[0]."\n<br />";
echo $xml->film->titre."\n<br />";
echo $xml->film[1]->titre."\n<br />";
echo $xml->film[1]->acteurs[0]->acteur[1]->nom."\n<br />";
echo $xml->film[1]->acteurs->acteur[1]->nom."\n<br />";
?>
```

En sortie nous avons donc:

```
Kill Bill: Vol. 2
Kill Bill: Vol. 2
Mais ou est donc passee la septieme compagnie ?
Pierre Mondy
Pierre Mondy
```

En fait, pour bien comprendre la structure de l'objet `$xml`, faisons le afficher:

Listing 15.14 : parse_videotheque_2.php

```
<?php
$xml = simplexml_load_file("videotheque.xml");
print_r($xml);
?>
```

Et le résultat:

```
SimpleXMLElement Object
(
  [film] => Array
    (
      [0] => SimpleXMLElement Object
        (
          [titre] => Kill Bill: Vol. 2
          [realisateur] => Quentin Tarantino
          [genres] => SimpleXMLElement Object
            (
              [genre] => Array
                (
                  [0] => action
                  [1] => drama
                  [2] => thriller
                )
            )
```

```
            [acteurs] => SimpleXMLElement Object
              (
                [acteur] => Array
                  (
                    [0] => SimpleXMLElement Object
                      (
                        [nom] => Uma Thurman
                        [role] => The Bride (Black Mamba)
                      )
                    [1] => SimpleXMLElement Object
                       (
                         [nom] => David Carradine
                         [role] => Bill (Snake Charmer)
                       )
                  )
              )
        )
    [1] => SimpleXMLElement Object
      (
        [titre] => Mais ou est donc passee la septieme compagnie ?
        [realisateur] => Robert Lamoureux
        [genres] => SimpleXMLElement Object
          (
            [genre] => Array
              (
                [0] => war
                [1] => action
                [2] => comedy
              )
          )
        [acteurs] => SimpleXMLElement Object
          (
            [acteur] => Array
              (
                [0] => SimpleXMLElement Object
                  (
                    [nom] => Jean Lefebvre
                    [role] => Pitivier
                  )
                [1] => SimpleXMLElement Object
                  (
                    [nom] => Pierre Mondy
                    [role] => Sergent Chaudard
                  )
              )
          )
      )
  )
)
```

Ce résultat nous apprend deux choses, d'une part, qu'il est facile d'accéder à n'importe quelle entité car la structure n'est ni plus moins qu'un tableau, et d'autre part, on apprend que dans le

cas d'un document XML imposant que la place mémoire occupée par ce tableau sera immense et donc simpleXML ne sera pas adapté pour de larges fichiers XML (De même que DOM).

Connaissant la structure de l'objet `$xml`, toute folie est désormais possible, par exemple lister les films ou les acteurs d'un film et désormais très facile, la preuve ci-après.

Listing 15.15 : parse_videotheque_3.php

```
<?php
$xml = simplexml_load_file("videotheque.xml");
foreach ($xml->film as $film) {
  echo "* ".$film->titre."\n<br />";
  foreach ($film->acteurs->acteur as $acteur) {
   echo " - ".$acteur->nom." joue dans le role de ".$acteur->role."\n<br />";
  }
}
?>
```

qui a pour résultat:

```
* Kill Bill: Vol. 2
 - Uma Thurman joue dans le role de The Bride (Black Mamba)
 - David Carradine joue dans le role de Bill (Snake Charmer)
* Mais ou est donc passee la septieme compagnie ?
 - Jean Lefebvre joue dans le role de Pitivier
 - Pierre Mondy joue dans le role de Sergent Chaudard
```

Simple mais nous n'avons pas vu comment récupérer un attribut. imdb est le numéro de film sachant que chaque film possède un numéro unique attribué par un organisme. On peut vouloir récupérer le numéro imdb du premier film, ou mieux encore, récupérer le titre du film dont on connaît le numéro. Voici comment il est possible de procéder :

Listing 15.16 : parse_videotheque_4.php

```
<?php
$xml = simplexml_load_file("videotheque.xml");
echo "Numero imdb du premier film: ".$xml->film[0]['imdb']."\n<br />";
echo "Numero imdb du second film: ".$xml->film[1]['imdb']."\n<br />";
foreach ($xml->film as $film) {
  if ((string)($film['imdb'])=="0378194")
    echo "Titre du film ayant 0378194 comme numero: ".$film->titre."\n<br />";
}
?>
```

Le résultat obtenu :

```
Numero imdb du premier film: 0378194
Numero imdb du second film: 0070356
Titre du film ayant 0378194 comme numero: Kill Bill: Vol. 2
```

En conclusion, récupérer un attribut n'est pas plus compliqué.

SimpleXML ne manque pas de gêne et se permet même d'utiliser les expressions XPath pour faire des recherches dans le document, pour cela il faut faire appel à la fonction `xpath()` de l'objet SimpleXML.

XPath est un ensemble de commandes de recherche très complet quoique rebutant au premier abord.


INTERNET

XPath
Présenter XPath ici serait trop long mais si vous ne connaissez pas ces expressions utiles pour faire des recherches dans un document, je vous conseille vivement de jeter un œil à:
http://xmlfr.org/w3c/TR/xpath/
pour la documentation de référence traduite en français
http://www.zvon.org/xxl/XPathTutorial/General_fre/examples.html
pour une aide rapide par l'exemple, ou encore le lien suivant qui met en parallèle requêtes SQL et leur équivalent utilisant XPATH.
http://www.fragbase.com/sql2xpath.php

L'exemple que nous avons pris étant simple, il ne va pas être facile de montrer les capacités de Xpath. Récupérer la liste des acteurs se fait très facilement en écrivant:

Listing 15.17 : parse_videotheque_5.php

```
<?php
$xml = simplexml_load_file("videotheque.xml");
foreach ($xml->xpath('//acteur') as $acteur) {
  echo $acteur->nom."\n<br />";
}
?>
```

Qui donne:

```
Uma Thurman
David Carradine
Jean Lefebvre
Pierre Mondy
```

Récupérer le film dont l'imdb est 0378194 se fait facilement également:

Listing 15.18 : parse_videotheque_6.php

```
<?php
$xml = simplexml_load_file("videotheque.xml");
$film = $xml->xpath('//film[@imdb=0378194]');
echo $film[0]->titre;
?>
```

Qui donne:

```
Kill Bill: Vol. 2
```

API

Nous avons vu très peu de fonctions dans les exemples précédents (deux pour être exact) mais nous pouvons déjà faire beaucoup. Il est possible d'analyser un fichier XML ou une chaîne de caractères XML avec `simplexml_load_file()` et `simplexml_load_string()` respectivement.

simplexml_load_file()

Transforme le fichier XML en objet facilement manipulable.

Syntaxe	`SimpleXMLElement simplexml_load_file(string $nomFichier)`
`$nomFichier`	Nom du fichier à charger.
`retour`	Objet de type SimpleXMLElement facilement manipulable.

simplexml_load_string()

Transforme la chaîne de caractères XML en objet facilement manipulable.

Syntaxe	`SimpleXMLElement simplexml_load_file(string $chaineXML)`
`$chaineXML`	Chaîne de caractères XML à charger.
`retour`	Objet de type SimpleXMLElement facilement manipulable.

Il est également possible de récupérer un objet SimpleXMLElement à partir d'un arbre DOM.

simplexml_import_dom()

Retourne un objet SimpleXMLElement à partir d'un arbre DOM.

Syntaxe	`SimpleXMLElement simplexml_import_dom(domNode $noeudDOM)`
`$noeudDOM`	Nœud récupéré par la méthode d'analyse syntaxique DOM.
`retour`	Objet de type SimpleXMLElement facilement manipulable.

Comme vu dans les exemples, `xpath()` est une fonction qui s'applique à un objet de type SimpleXMLElement obtenu par une des trois fonctions précédentes.

SimpleXMLElement->xpath()

Retourne un tableau d'objet de type SimpleXMLElement correspondant à la recherche XPath.

Syntaxe	`array[SimpleXMLElement] xpath(string $xpath)`
`$requete`	Requête XPath.
`retour`	Tableau d'objets SimpleXMLElement.

SimpleXMLElement->children()

Retourne un objet de type SimpleXMLElement correspondant aux enfants du nœud courant.

Syntaxe	`SimpleXMLElement children()`
`retour`	Un objet de type SimpleXMLElement.

Cette dernière fonction permet de récupérer les "enfants" d'un nœud, voici un exemple d'utilisation:

Listing 15.19 : parse_videotheque_7.php

```
<?php
$xml = simplexml_load_file("videotheque.xml");
foreach ($xml -> children() as $test) {
    echo $test->titre."\n<br />";
}
?>
```

Dont le résultat attendu est bien évidement:

```
Kill Bill: Vol. 2
Mais ou est donc passee la septieme compagnie ?
```

SimpleXMLElement->attributes()

Retourne un objet de type SimpleXMLElement correspondant aux attributs du nœud courant.

Syntaxe	`SimpleXMLElement attributes()`
`retour`	Un objet de type SimpleXMLElement.

Voici un court exemple d'utilisation:

Listing 15.20 : parse_videotheque_8.php

```
<?php
$xml = simplexml_load_file("videotheque.xml");
foreach ($xml->film[0]->attributes() as $attribut => $valeur) {
    echo $attribut."=".$valeur."\n<br />";
}
?>
```

Dont le résultat attendu est:

```
imdb=0378194
```

Bien entendu, rien n'empêche de modifier les objets `SimpleXMLElement`. Mais une fois modifiés, on peut vouloir récupérer le contenu de ces objets en format XML pour l'écrire par exemple dans un fichier. Pour cela, il suffira alors d'utiliser la fonction `asXML()` qui s'applique à un objet de type `simpleXMLElement`.

SimpleXMLElement->asXML()

Retourne une chaîne de caractères XML valide représentant l'objet `SimpleXMLElement`.

Syntaxe	`string asXML()`
`retour`	Une chaîne XML

Listing 15.21 : parse_videotheque_9.php

```
<?php
$xml = simplexml_load_file("videotheque.xml");
$xml->film[0]->titre = "Kill Bill Volume 2";
echo $xml->asXML();
?>
```

Et en retour on a le XML modifié (le titre de Kill Bill: Vol. 2 a été subtilement modifié)"

```
<?xml version="1.0" standalone="yes"?>
<videotheque>
  <film imdb="0378194">
    <titre>TITRE</titre>
    <realisateur>Quentin Tarantino</realisateur>
    <genres>
      <genre>action</genre>
      <genre>drama</genre>
      <genre>thriller</genre>
    </genres>
    <acteurs>
      <acteur>
        <nom>Uma Thurman</nom>
```

```
        <role>The Bride (Black Mamba)</role>
      </acteur>
      <acteur>
        <nom>David Carradine</nom>
        <role>Bill (Snake Charmer)</role>
      </acteur>
    </acteurs>
  </film>
  <film imdb="0070356">
    <titre>Mais ou est donc passee la septieme compagnie ?</titre>
    <realisateur>Robert Lamoureux</realisateur>
    <genres>
      <genre>war</genre>
      <genre>action</genre>
      <genre>comedy</genre>
    </genres>
    <acteurs>
      <acteur>
        <nom>Jean Lefebvre</nom>
        <role>Pitivier</role>
      </acteur>
      <acteur>
        <nom>Pierre Mondy</nom>
        <role>Sergent Chaudard</role>
      </acteur>
    </acteurs>
  </film>
</videotheque>
```

Et voilà pour la bibliothèque simpleXML, en tout et pour tout il y a sept fonctions utiles vous permettant de lire/modifier de petits documents XML très simplement !

15.4. XSLT

Présentation

La transformation XSL s'appuie sur un fichier décrivant ce que la transformation doit retourner lorsque telle ou telle balise est rencontrée. Nous supposerons ici que vous êtes familier avec ce genre de transformation.

XML, XSL
Vous trouverez plus d'informations sur XML et la XSL, sur le site :
http://www.xmlfacile.com
sans oublier la référence (en anglais) :
http://www.w3.org/Style/XSL/
et sa traduction :
http://www.xmlfr.org/w3c/TR/xslt/

Par la suite, à titre d'exemple, nous utiliserons la feuille de styles suivante :

Listing 15.22 : mediatheque_01.xsl

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xsl:transform xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="html" />

<xsl:template match="/">
    <html>
        <body>
            <xsl:apply-templates />
        </body>
    </html>
</xsl:template>

<xsl:template match="mediatheque">
    <h1>Contenu de la médiathèque</h1>
    <table border="1">
        <xsl:apply-templates select="cdaudio" />
    </table>
</xsl:template>

<xsl:template match="cdaudio">
    <tr><td colspan="2" bgcolor="#BBBBFF">
             <b><xsl:value-of select="@interprete" /></b>
     </td></tr>
     <tr><td colspan="2" bgcolor="#DDDDFF">
             <b><i><xsl:value-of select="@titre" /></i></b>
     </td></tr>

     <xsl:apply-templates select="chanson" />
</xsl:template>

<xsl:template match="chanson">
    <tr>
        <td><xsl:apply-templates select="titre" /></td>
        <td><xsl:apply-templates select="duree" /></td>
    </tr>
</xsl:template>

<xsl:template match="titre">
    <xsl:value-of select="text()" />
</xsl:template>

<xsl:template match="duree">
    <xsl:value-of select="text()" />
</xsl:template>

<xsl:template match="*">
</xsl:template>

</xsl:transform>
```

Les transformations de documents XML

La bibliothèque XSL

PHP 4
Cette bibliothèque a été introduite dans la version 5 de PHP, elle n'est donc pas disponible pour PHP4. Vous devrez donc dans ce cas utiliser la bibliothèque XSLT décrite plus loin.

Erreur de jeunesse ?
Cette bibliothèque PHP étant récente, elle peut être amenée à évoluer et est donc marquée comme étant expérimentale.

Installation

Sous Windows

Que ce soit avec l'archive de PHP Group ou avec EasyPHP, vous devrez vous assurer que vous possédez bien un fichier *php_xsl.dll* dans le répertoire des extensions PHP. Il vous suffit alors d'ajouter au fichier *php.ini* ou de décommenter une ligne

```
extension=php_xsl.dll
```

Sous Linux

Il vous faut d'abord récupérer et installer la bibliothèque `libxslt` disponible à l'adresse ftp://xmlsoft.org comme indiqué sur http://xmlsoft.org/XSLT. en tapant les commandes désormais habituelles:

```
# tar zxvf libxslt-1.1.10.tar.gz
# ./configure --prefix=/usr/local
# make
# make install
```

Problème de compilateur (fails sanity check)
Il se trouve que l'opération `configure` *ne s'est pas déroulé correctement dans notre environnement. Celui-ci n'acceptant pas apparemment pas notre compilateur. Nous ne nous sommes alors pas attardés sur le problème et nous nous en sommes remis aux "packages" de notre Debian. Nous avons donc remplacé l'installation précédente par*

```
# apt-get install libxslt1-dev
```

Une fois `libxslt` installé, il suffit de recompiler PHP avec l'option de configuration `"--with-xsl=/usr/local"`.

Vous pouvez vous reporter au chapitre "Installation" pour plus de détails sur la compilation de PHP.

Vérification

Comme vous en avez pris l'habitude, vous pouvez vérifier que l'installation s'est déroulée correctement en appelant un script contenant `<?php phpinfo(); ?>` qui doit afficher :

xsl

XSL	enabled
libxslt Version	1.1.8
libxslt compiled against libxml Version	2.6.11
EXSLT	enabled
libexslt Version	1.1.8

Figure 15.5 : *phpinfo()*

Utilisation

La transformation d'un document XML via XSLT est très simple. Il suffit de charger les documents XML et XSL dans des objets DOMDocument, d'instancier un objet XSLTProcessor, de charger le document XSL et d'appeler la méthode de transformation.

Vous pouvez vous reporter au chapitre "Le parseur DOM" pour plus de détails sur l'objet DOMDocument.

XSLTProcessor();

Objet permettant les transformations XSLT

Syntaxe : `XSLTProcessor()`

XSLTProcessor->importStyleSheet()

Définit la feuille de style à utiliser pour les prochaines transformations XSL.

Syntaxe : `boolean importStyleSheet(DOMDocument $documentXSL)`

`$documentXSL`	Document DOM contenant la feuille de style
`retour`	`FALSE` en cas d'échec, `NULL` sinon

XSLTProcessor->transformToXML()

Retourne, sous forme d'une chaîne de caractères, le résultat de l'application de la feuille de style précédemment sélectionnée sur le document XML indiqué.

Syntaxe :	`string transformToXML(DOMDocument $documentXML)`
`$documentXML`	Document DOM contenant le document XML à transformer.
`retour`	Résultat de la transformation ou `FALSE` en cas d'erreur.

XSLTProcessor->transformToDoc()

Retourne, sous forme d'un DOMDocument, le résultat de l'application de la feuille de style précédemment sélectionnée sur le document XML indiqué.

Syntaxe :	`DOMDocument transformToDoc(DOMDocument $documentXML)`
`$documentXML`	Document DOM contenant le document XML à transformer.
`retour`	Résultat de la transformation sous forme de DOMDocument ou `FALSE` en cas d'erreur.

Listing 15.23 : xsl_01.php

```
<?php
    $cheminAbsolu = dirname(__FILE__);
    // Chargement du document XML
    $xmlDoc = new DOMDocument();
    $xmlDoc->load("$cheminAbsolu/../src_xml/mediatheque_01b.xml");

    $xslDoc = new DOMDocument();
    $xslDoc->load("$cheminAbsolu/../src_xml/mediatheque_01.xsl");

    // Chargement de la feuille de style
    $xslt = new XSLTProcessor();
    $xslt->importStyleSheet($xslDoc);

    // Transformation du fichier XML via XSLT
    // et envoi du résultat directement à l'écran
    echo $xslt->transformToXML($xmlDoc);
?>
```

Le résultat obtenu sera alors :

Figure 15.6 : *Exemple de transformation XSL*

Dans l'ensemble la transformation s'est relativement correctement bien passée. Néanmoins les entités externes n'ont pas été prises en compte (`&nc;` n'a par exemple pas été remplacé par "Non Communiqué". Il est trop tôt pour dire s'il s'agit d'un bug ou bien s'il faut attendre que cette bibliothèque évolue encore un peu pour répondre totalement à nos besoins.

La bibliothèque XSLT/Sablotron

PHP 5
Cette bibliothèque n'est pas disponible avec PHP 5.X. Elle a été remplacée par la bibliothèque XSL s'appuyant sur `libxslt`.

Installation

Sous Windows

Que ce soit avec l'archive de PHP Group ou avec EasyPHP, vous devrez vous assurer que vous possédez bien un fichier *php_xslt.dll* dans le répertoire des extensions PHP. Il vous suffit alors d'ajouter au fichier *php.ini* ou de décommenter une ligne

```
extension=php_xslt.dll
```

Sous Linux

Pour pouvoir utiliser la bibliothèque `XSLT` avec Sablotron, vous devez au préalable récupérer la bibliothèque `Sablotron`. Celle-ci est disponible sur le site **http://www.gingerall.com/** ainsi que sur le CD-ROM fourni (sous le nom *Sablot-1.0.tar.gz*). Cette bibliothèque nécessite également la bibliothèque `expat`. Cette dernière est disponible à l'adresse **http://sourceforge.net/projects/expat/** ainsi que sur le CD-ROM fourni (sous le nom *expat-1.95.6.tar.gz*).

Installation d'expat

Copiez le fichier *expat-1.95.6.tar.gz* sous le répertoire */usr/local/src/lib* (par exemple), puis saisissez les commandes :

```
# gunzip expat-1.95.6.tar.gz
# tar xvf expat-1.95.6
```

Le fichier est à présent décompressé.

```
# cd expat-1.95.6
# ./configure
# make
# make install
```

La bibliothèque `expat` est maintenant disponible sous */usr/local/lib* (fichiers *libexpat**).

Installation de Sablotron

Avant d'installer Sablotron vous aurez besoin de patcher un des fichiers installés par expat (si comme indiqué vous utilisez la version 1.95.6). Pour cela, allez dans le répertoire */usr/local/include* et éditez le fichier *expat.h*. Recherchez le bout de code suivant:

```
enum XML_Status {
  XML_STATUS_ERROR = 0,
#define XML_STATUS_ERROR XML_STATUS_ERROR
  XML_STATUS_OK = 1
#define XML_STATUS_OK XML_STATUS_OK
};
```

et déplacez le en début de fichier (avant le bloc "`enum XML_Error {`" par exemple)

Maintenant vous pouvez copier le fichier *Sablot-1.0.tar.gz* sous le répertoire */usr/local/src/lib* (par exemple), puis saisir les commandes :

```
# gunzip Sablot-1.0.tar.gz
# tar xvf Sablot-1.0.tar
# cd Sablot-1.0
```

Le fichier est à présent décompressé.

```
# cd Sablot-1.0
# export SABLOT_GPL=1
# ./configure
# make
# make install
```

SABLOT_GPL
La commande `export SABLOT_GPL=1` est nécessaire si vous souhaitez utiliser la fonction `xslt_set_encoding()`.

La bibliothèque `Sablotron` est désormais disponible sous */usr/local/lib* (fichiers *libsab**) ainsi que le fichier */usr/local/bin/sabcmd*.

Maintenant que `Sablotron` est installé, il suffit de l'intégrer à PHP, en le recompilant avec l'option de configuration "`--enable-xslt --with-xslt-sablot`". Mais attention! Avec la version 3 de gcc, vous risquez d'avoir des petits problèmes. Pour y palier, vous devrez lancer la commande suivante avant l'appel à la commande `./configure`.

```
# export LDFLAGS="-lstdc++"
```


RENVOI

Vous pouvez vous reporter au chapitre "Installation" pour plus de détails sur la compilation de PHP.


REMARQUE

Apache
Le serveur Apache intègre sa propre version d'`expat` qui peut entrer en conflit avec celle qui vient d'être installée. Cependant, depuis la version 1.3.21 d'Apache, il suffit de recompiler Apache pour qu'il intègre la version d'`expat` qui vient d'être installée. Si vous disposez d'une version plus ancienne d'Apache pensez à la mettre à jour.

Vérification

Comme vous en avez pris l'habitude, vous pouvez vérifier que l'installation s'est déroulée correctement en appelant un script contenant `<?php phpinfo(); ?>` qui doit afficher :

xslt

XSLT support	enabled
Backend	Sablotron
Sablotron Version	1.0
Sablotron Information	Cflags: -g -O2 Libs: -L/usr/local/lib -lexpat Prefix: /usr/local

Figure 15.7 : *phpinfo()*

Utilisation

La transformation d'un document XML via XSLT se réalise en trois étapes :

- Création d'un analyseur XSLT ;
- Transformation du document ;
- Libération des ressources occupées par l'analyseur.

Ceci se traduit par l'appel aux fonctions :

- `xslt_create()` ;
- `xslt_process()` ;
- `xslt_free()`.

xslt_create()

Crée un nouvel analyseur XSLT.

Syntaxe	`resource xslt_create(void)`
`retour`	Référence sur un analyseur XSLT.

xslt_process()

Applique une transformation XSL sur un document XML.

Syntaxe	`mixed xslt_process(resource $analyseurXSLT, string $uriXML, string $uriXSL [, string $uriResultat, [array $parametresAnalyseur, [array $parametresXSL]]])`
`$analyseurXSLT`	Référence sur un analyseur XSLT tel que retourné par la fonction `xslt_create()`.
`$uriXML`	URI (adresse) du fichier XSL. Ce doit être soit un chemin relatif au serveur (et non pas relatif au script PHP !) soit un chemin absolu précédé de "file://", "http://", etc. selon la localisation du fichier.
`$uriXSL`	URI (adresse) du fichier XSL. Ce doit être soit un chemin relatif au serveur (et non pas relatif au script PHP !) soit un chemin absolu précédé de "file://", "http://", etc. selon la localisation du fichier.
`$uriResultat`	URI (adresse) où doit être écrit le fichier résultat. Ce doit être soit un chemin relatif au serveur (et non pas relatif au script PHP !) soit un chemin absolu précédé de "file://" selon la localisation du fichier. Si ce paramètre n'est pas spécifié, alors le résultat est donné dans la valeur retour de la fonction.
`$parametresXSLT`	Qui doit contenir un tableau associatif avec pour clés les paramètres utilisés par la feuille XSLT et pour valeurs les valeurs de ces différents paramètres.
`$parametresAnalyseur`	Qui doit contenir un tableau présentant les paramètres à passer à l'analyseur Sablotron.
`retour`	Chaîne de caractères contenant le résultat de la transformation si aucun nom de fichier n'est précisé, `TRUE` si le résultat a été sauvegardé dans un fichier avec succès, `FALSE` sinon.


REMARQUE

Les encodages supportés en entrée

Avec la bibliothèque `Sablotron`, les fichiers XML peuvent être encodés en "UTF-8", "UTF-16", "ASCII", "ISO-8859-1", "ISO-8859-2" et "Windows-1250". Mais cette liste est plus importante si la bibliothèque `iconv` est disponible dans l'environnement de l'analyseur XSLT.

xslt_free()

Libère les ressources occupées par l'analyseur.

Syntaxe `void xslt_free(resource $analyseurXSLT)`

`$analyseurXSLT` Référence sur un analyseur XSLT tel que retourné par la fonction `xslt_create()`.

Listing 15.24 : xslt_01.php

```
<?php
    // Création d'un nouvel analyseur XSLT
    $xslt = xslt_create();

    // Transformation du fichier XML via XSLT
    // et envoi du résultat directement à l'écran
    $cheminAbsolu = "file://".dirname(__FILE__);
    echo xslt_process($xslt, "$cheminAbsolu/../src_xml/mediatheque_01b.xml",
                             "$cheminAbsolu/../src_xml/mediatheque_01.xsl");

    // Libération des ressources
    xslt_free($xslt);
?>
```

Le résultat obtenu sera alors :

Contenu de la médiathèque

Noir Désir	
des Visages des Figures	
L'enfant roi	6:03
Le grand incendie	4:37
Le vent nous portera	4:48
Placebo	
Without You I'm Nothing	
Pure Morning	4:15
Without You I'm Nothing	4:30
Every You, Every Me	5:00
Radiohead	
OK Computer	
Karma Police	4:23
Muse	
Showbiz	
Feeling Good	Non communiqué

Figure 15.8 : *Exemple de transformation XSL*

Vous constaterez que, dans ce cas, les références externes sont prises en compte automatiquement (ce qui n'est pas le cas avec le parseur expat).

Le résultat est encodé en UTF-8 (ce qui fait que les caractères accentués pourraient avoir une allure étrange selon la configuration de votre navigateur). Si vous souhaitez un résultat en ISO-8859-1 (ou autre) vous devez, au préalable, configurer l'analyseur.

Configuration de l'analyseur

Par défaut, le document créé est codé en UTF-8, mais il est possible de choisir un type d'encodage alternatif avec `xslt_set_encoding()`.

xslt_set_encoding()

Précise l'encodage du document généré (cette fonction n'est pas disponible dans les versions Windows de PHP sorties jusque-là, à savoir 4.2.1).

Syntaxe	`void xslt_set_encoding(resource $analyseurXSLT, string $encodage)`
`$analyseurXSLT`	Référence sur un analyseur XSLT tel que retourné par la fonction `xslt_create()`.
`$encodage`	Encodage choisi (ex. : "ISO-8859-1", "UTF-8", ...).

Encodages supportés en sortie
La liste des encodages supportés dépend de l'environnement. Si la bibliothèque `iconv` est présente alors `Sablotron` supporte tous les encodages que supporte `iconv`.

xslt_set_base()

Précise l'URI de base des documents précisés par un chemin relatif.

Syntaxe	`void xslt_set_base(resource $analyseurXSLT, string $baseURI)`
`$analyseurXSLT`	Référence sur un analyseur XSLT tel que retourné par la fonction `xslt_create()`.
`$baseURI`	URI de base pour les chemins relatifs.

Listing 15.25 : xslt_02.php

```
<?php
    // Création d'un nouvel analyseur XSLT
    $xslt=xslt_create();

    // Selection d'un encodage de sortie ISO-8859-1
    $status = xslt_set_encoding($xslt, "ISO-8859-1");
```

```
    // Précision du chemin du repertoire de base
    $cheminAbsolu="file://".dirname(__FILE__);
    xslt_set_base($xslt, "$cheminAbsolu/../src_xml/");

    echo xslt_process($xslt, "mediatheque_01b.xml",
                             "mediatheque_01.xsl");

    // Libération des ressources
    xslt_free($xslt);
?>
```

Gestion des erreurs

Il est évidemment possible à tout moment de connaître quelle a été la dernière erreur rencontrée, soit sous la forme d'un code avec `xslt_errno()`, soit sous la forme d'un message d'erreur en anglais avec `xslt_error()`.

xslt_errno()

Retourne le code de la dernière erreur rencontrée par l'analyseur XSLT.

Syntaxe	`int xslt_errno(resource $analyseurXSLT)`
`$analyseurXSLT`	Référence sur un analyseur XSLT tel que retourné par la fonction `xslt_create()`.
`retour`	Code d'erreur.

xslt_error()

Retourne le message d'erreur correspondant à la dernière erreur rencontrée par l'analyseur XSLT.

Syntaxe	`string sxlt_error(resource $analyseurXSLT)`
`$analyseurXSLT`	Référence sur un analyseur XSLT tel que retourné par la fonction `xslt_create()`.
`retour`	Message d'erreur.

Il est également (en théorie) possible de faire exécuter sa propre fonction lorsqu'une erreur est rencontrée. Pour cela, il faut, au préalable, déclarer la fonction auprès de l'analyseur XSLT avec la fonction `xslt_set_error_handler()`.

xslt_set_error_handler()

Déclare auprès de l'analyseur XSLT une fonction de gestion des erreurs.

Syntaxe	`void xslt_error_handler(resource $analyseurXSLT, string $fonctionErreur)`
`$analyseurXSLT`	Référence sur un analyseur XSLT tel que retourné par la fonction `xslt_create()`.
`$fonctionErreur`	Nom de la fonction devant gérer les erreurs.

La fonction de gestion des erreurs (`$fonctionErreur`) aura la syntaxe suivante :

Syntaxe	`void maFonction(resource $analyseurXSLT, int $niveauErreur, int $erreur, array $info)`
`$analyseurXSLT`	Référence sur un analyseur XSLT telle que retournée par la fonction `xslt_create()`.
`$niveauErreur`	Le niveau d'erreur (les niveaux PHP ?).
`$erreur`	On croirait le code d'erreur précédent.
`$info`	Tableau associatif contenant diverses informations sur l'erreur. Vous trouverez parmi les clés : "`msgtype`" associé à la valeur "`error`". "`code`" : le code d'erreur. "`module`" : le module ayant détecté l'erreur (i.e. "Sablotron"). "`URI`" : l'URI du document en cause. "`line`" : la ligne à laquelle l'erreur a été détectée. "`msg`" : le message d'erreur.

Bug

Dans une précédente version testée (PHP 4.2.1 + Sablot-0.90 + Expat 1.95.2) les noms des clés étaient souvent "complétés" par des signes cabalistiques. Cela semble désormais corrigé (PHP 4.3.2 + Sablot-1.0 + Expat 1.95.6).

Gestion des traces

Il est possible de suivre à la trace le travail de l'analyseur XSLT grâce à la fonction `xslt_set_log()`. La mise en œuvre de ce mécanisme de trace se déroule en deux temps. Premier temps : autoriser les traces, et second temps : indiquer dans quel fichier stocker les traces (par défaut les traces vont sur la sortie `stderr`).

xslt_set_log()

Demande l'activation/l'arrêt des traces. Indique quel fichier utiliser pour les traces.

Syntaxe	`void xslt_set_log(resource analyseurXSLT, mixed $onOffNomFichier)`
`$analyseurXSLT`	Référence sur un analyseur XSLT telle que retournée par la fonction `xslt_create()`.
`$onOffNomFichier`	`TRUE` si vous souhaitez activer les traces, `FALSE` si vous souhaitez stopper les traces, ou bien encore nom de fichier (il ne s'agit pas ici d'une URI, donc pas besoin de le faire précéder de "file://") vers lequel vous souhaitez que les traces soient redirigées.

15.5. Génération de messages XML

Depuis son arrivée, XML est utilisé dans de nombreux domaines. Cela devient l'élément de base du stockage et de l'échange de l'information. Pas étonnant de voir de nouvelles normes apparaître pour l'échange de messages entre serveurs.

Parmi elles, l'on trouve la norme WDDX (Web Distributed Data Exchange, c'est-à-dire la norme d'échange de données sur le Web). Celle-ci est destinée à échanger des structures de données complexes entre les langages de programmation.


REMARQUE

Web services
Le langage XML est également à la base de l'échange d'informations des "Web services".

Les messages WDDX

Installation

Sous Windows

Que ce soit avec l'archive du PHP Group ou EasyPHP, le support de WDDX est activé d'office.

Sous Linux

Il suffit de recompiler PHP avec l'option de configuration "`--enable-wddx`".


RENVOI

Vous pouvez vous reporter au chapitre "Installation" pour plus de détails sur la compilation de PHP.

Vérification

Là encore, vous pouvez vérifier que WDDX est bien activé en appelant un script contenant `<?php phpinfo(); ?>` et qui devra cette fois afficher :

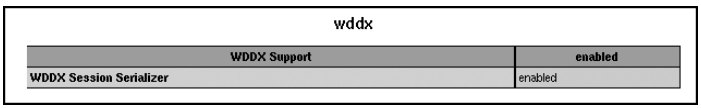


Figure 15.9 : *phpinfo()*

Utilisation

L'utilisation de la bibliothèque `WDDX` est très simple. Créer un paquet WDDX décrivant le contenu d'une variable consiste à appeler la fonction `wddx_serialize_value()`.

wddx_serialize_value()

Crée un paquet WDDX décrivant le contenu d'une variable unique.

Syntaxe	`string wddx_serialize_value(mixed $variable [, string $commentaire])`
`$variable`	Variable à décrire dans le paquet WDDX.
`$commentaire`	Commentaire à inclure dans l'attribut "`comment`" de la balise "`header`" du paquet WDDX.
`retour`	La chaîne de caractères contenant le paquet WDDX.

Exemple d'utilisation :

Listing 15.26 : wddx_01.php

```
<?php
  include("wddx2html_inc.php");

  $variable1  = "Du Texte";
  $paquetWDDX = wddx_serialize_value($variable1,
                        "Test avec chaîne de caractères");

  echo wddx2html($paquetWDDX);

  echo "<br />";

  $tab = array ("Test", 2, array("Cle"=>"Valeur"));
  $paquetWDDX = wddx_serialize_value($tab, "Test avec tableau");

  echo wddx2html($paquetWDDX);

?>
```

Ce script utilise une fonction (faite maison) afin de rendre l'affichage du résultat plus lisible depuis un navigateur.

Listing 15.27 : wddx2html_inc.php

```
<?php

    // Retourne un paquetWDDX avec une forme
    // permettant une meilleure lecture en HTML.

    function wddx2html($paquetWDDX)
    {
        $resultat = "";
        $tab      = "";

        $balises = explode("<", $paquetWDDX);
        for ($i=0; $i<count($balises); $i++) {
          if ($balises[$i]!="") {
            // S'il s'agit d'une balise fermante
            // indenter un peu moins la ligne
            if (eregi("^/",$balises[$i]))
               $tab = substr($tab, 0, -12);

            $resultat .= $tab . htmlspecialchars("<".$balises[$i])."<br />";

            // S'il s'agit d'une balise ouvrante
            // indenter un peu plus la ligne
            // suivante
            if ((!eregi("^/",$balises[$i]))&&
                (!eregi("/>$",$balises[$i])))
               $tab .= "  ";
          }
        }
        return $resultat;
    }
?>
```

dont voici le résultat :

```
<wddxPacket version='1.0'>
<header>
<comment>Test avec chaîne de caractères
</comment>
</header>
<data>
<string>Du Texte
</string>
</data>
</wddxPacket>

<wddxPacket version='1.0'>
  <header>
    <comment>Test avec tableau
    </comment>
  </header>
```

```
  <data>
    <array length='3'>
      <string>Test
      </string>
      <number>2
      </number>
      <struct>
        <var name='Cle'>
          <string>Valeur
          </string>
        </var>
      </struct>
    </array>
  </data>
</wddxPacket>
```

Pour créer un paquet WDDX décrivant plusieurs variables, il faut faire appel à `wddx_serialize_vars()`.

wddx_serialize_vars()

Crée un paquet WDDX décrivant plusieurs variables.

Syntaxe `string wddx_serialize_vars(mixed $nomVariable [, mixed $nomVariable...])`

`$nomVariable` Nom de la variable ou tableau de noms de variables.

Listing 15.28 : wddx_02.php

```
<?php
  include("wddx2html_inc.php");

  $tableau = array ("Test", 2, array("Cle"=>"Valeur"));

  $variable1 = "Variable1";
  $variable2 = "Variable2";
  $tableauNom = array ("variable1", "variable2");

  $paquetWDDX = wddx_serialize_vars("variable1", "tableau", $tableauNom);

  echo wddx2html($paquetWDDX);

?>
```

affichera :

```
<wddxPacket version='1.0'>
 <header/>
```

```
<data>
 <struct>
  <var name='variable1'>
   <string>Variable1
   </string>
  </var>
  <var name='tableau'>
   <array length='3'>
    <string>Test
    </string>
    <number>2
    </number>
    <struct>
     <var name='Cle'>
      <string>Valeur
      </string>
     </var>
    </struct>
   </array>
  </var>
  <var name='variable1'>
   <string>Variable1
   </string>
  </var>
  <var name='variable2'>
   <string>Variable2
   </string>
  </var>
 </struct>
</data>
</wddxPacket>
```

Dans ce cas, il n'est pas possible de spécifier un attribut "`comment`".

Si vous souhaitez spécifier un attribut "`comment`", et surtout si vous souhaitez créer un packet WDDX en ajoutant les variables les unes après les autres, vous pouvez utiliser les fonctions `wddx_packet_start()`, `wddx_add_vars()` et `wddx_packet_end()`.

wddx_packet_start()

Commence un paquet WDDX.

Syntaxe	`int wddx_packet_start([string $comment])`
`$comment`	Commentaire à inclure dans l'attribut "`comment`" de la balise "`header`".
`retour`	Référence sur un paquet WDDX.

wddx_add_vars()

Ajoute des variables à un paquet WDDX.

Syntaxe	`void wddx_add_vars(resource $paquetWDDX, mixed $nomVariable [,mixed $nomVariable ...])`
`$paquetWDDX`	Référence sur un paquet WDDX telle que retournée par `wddx_packet_start()`.
`$nomVariable`	Nom de la variable, tableau de noms de variables.

wddx_packet_end()

Clôt et retourne le paquet WDDX.

Syntaxe	`string wddx_packet_end(int $paquetWDDX)`
`$paquetWDDX`	Référence sur un paquet WDDX telle que retournée par `wddx_packet_start()`.
`retour`	La chaîne de caractères contenant le paquet WDDX.

Listing 15.29 : wddx_03.php

```
<?php
  include("wddx2html_inc.php");

  $tableau = array ("Test", 2, array("Cle"=>"Valeur"));

  $variable1 = "Variable1";
  $variable2 = "Variable2";
  $tableauNom = array ("variable1", "variable2");

  $refWDDX = wddx_packet_start("Mon commentaire");
  wddx_add_vars($refWDDX, "variable1", "tableau");
  wddx_add_vars($refWDDX, $tableauNom);
  $paquetWDDX = wddx_packet_end($refWDDX);

  echo wddx2html($paquetWDDX);
?>
```

Enfin, fort heureusement, la bibliothèque `WDDX` ne permet pas seulement de générer des paquets WDDX ; elle permet aussi de les lire. Pour cela, vous disposez de la fonction `wddx_deserialize()`.

wddx_deserialize()

Lit un paquet WDDX.

Syntaxe	`mixed wddx_deserialize(string $paquetWDDX)`
`$paquetWDDX`	Paquet WDDX à lire.
`retour`	Retourne, selon les cas, un nombre, une chaîne de caractères, un tableau associatif ayant pour clé le nom de la variable et pour valeur la valeur de la variable. Ce tableau pourra être très simplement converti en une série de variables grâce à la fonction `extract()` présentée dans le chapitre concernant les tableaux.

Listing 15.30 : wddx_04.php

```
<?php

   echo "<b>Désérialisation d'un paquet WDDX de type tableau ";
   echo "(issu des exemples précédents)</b>";
   echo "<br />";

   $paquetWDDX  = "<wddxPacket version='1.0'>";
   $paquetWDDX .= "<header><comment>Mon commentaire</comment></header>";
   $paquetWDDX .= "<data><struct>";
   $paquetWDDX .= "<var name='variable1'><string>Variable1</string></var>";
   $paquetWDDX .= "<var name='tableau'><array length='3'>";
   $paquetWDDX .= "<string>Test</string><number>2</number>";
   $paquetWDDX .= "<struct>";
   $paquetWDDX .= "<var name='Cle'><string>Valeur</string></var>";
   $paquetWDDX .= "</struct>";
   $paquetWDDX .= "</array></var>";
   $paquetWDDX .= "<var name='variable1'><string>Variable1</string></var>";
   $paquetWDDX .= "<var name='variable2'><string>Variable2</string></var>";
   $paquetWDDX .= "</struct></data></wddxPacket>";

   $tableauWDDX = wddx_deserialize($paquetWDDX);

   if (is_array($tableauWDDX)) extract($tableauWDDX);

   echo "variable1 = $variable1<br />";
   echo "tableau   = ";
   print_r($tableau);
   echo "<br/>";
   echo "variable2 = $variable2<br />";


   echo "<b>Cas d'une variable de type chaîne de caractères</b><br />";
   $paquetWDDX = wddx_serialize_value("Texte");
   echo wddx_deserialize($paquetWDDX)."<br />";

   echo "<b>Cas d'une variable de type nombre</b><br />";
```

```
    $paquetWDDX = wddx_serialize_value(12);
    echo wddx_deserialize($paquetWDDX)."<br />";

?>
```

aura bien l'effet attendu, en retournant :

```
Désérialisation d'un paquet WDDX de type tableau (issu des exemples précédents)
variable1 = Variable1
tableau = Array ( [0] => Test [1] => 2 [2] => Array ( [Cle] => Valeur ) )
variable2 = Variable2
Cas d'une variable de type chaîne de caractères
Texte
Cas d'une variable de type nombre
12
```

Chapitre 16

La gestion des protocoles HTTP, FTP, SOAP, etc.

16.1. Fonctions réseau (de base)

16.2. Réseau

Les fonctions présentées dans ce chapitre ne permettent pas de réels développements, mais constituent une "trousse à outils" parfois nécessaire dans la manipulation d'informations réseau (comme ce peut être le cas, par exemple, avec l'utilisation des sockets). Elles ne nécessitent aucune installation particulière.

Adresses IP et DNS

L'opération la plus souvent sollicitée est certainement celle qui consiste à déterminer l'adresse IP d'une machine lorsque l'on ne connaît que son nom. Pour cela, vous disposez de la fonction `getHostByName()`.

---

getHostByName()

Retourne l'adresse IP de la machine précisée par son nom.

Syntaxe	`string getHostByName(string $nomMachine)`
`$nomMachine`	Nom de la machine.
`retour`	Adresse IP de la machine, ou `$nomMachine` si aucune adresse IP n'a pu être trouvée.

Ainsi, le code suivant :

```
<?php
   echo getHostByName("localhost");
?>
```

retournera très probablement :

```
127.0.0.1
```

Alors que :

```
<?php
   echo getHostByName("www.php.net");
?>
```

retournera une adresse IP publique comme, par exemple,

```
208.210.50.161
```

Il est également possible de réaliser l'opération inverse, à savoir récupérer un nom de machine à partir de son adresse IP grâce à `getHostByAddr()`.

getHostByAddr()

Retourne un nom de machine associé à l'adresse IP précisée (tel qu'on le trouve dans le fichier */etc/hosts* des sytèmes UNIX/Linux).

Syntaxe	`string getHostByAddr(string $adresseIP)`
`$adresseIP`	Adresse IP de la machine.
`retour`	Nom associé à la machine, ou `$adresseIP` si aucun nom n'a pu être trouvé.

Le code suivant :

```
<?php
    echo getHostByAddr("127.0.0.1");
?>
```

pourra retourner :

```
localdomain.localhost
```

On notera au passage que l'opération n'est pas nécessairement réversible. Si `getHostByName("localhost")` retourne "127.0.0.1" cela n'implique pas que `getHostByAddr("127.0.0.1")` retourne "localhost". En effet, une machine peut avoir plusieurs noms (via des alias), et c'est donc le nom principal qui est retourné par `getHostByAddr()`.

De même, une unique machine peut posséder plusieurs adresses IP. Pour en déterminer la liste, vous pouvez faire appel à `getHostByNameL()`.

getHostByNameL()

Retourne la liste des adresses IP de la machine précisée par son nom.

Syntaxe	`array getHostByNameL($nomMachine)`
`$nomMachine`	Nom de la machine.
`retour`	Tableau indexé des adresses IP de la machine, ou `FALSE` si aucune adresse IP n'a pu être trouvée.

PHP dispose de fonctions permettant de convertir des adresses IP précisées sous la forme "classique" xxx.xxx.xxx.xxx en adresses sous la forme d'entiers, et réciproquement.

ip2long()

Convertit une adresse IP du format "xxx.xxx.xxx.xxx" en un entier.

Syntaxe :	`int ip2long(string $adresseIP)`
`$adresseIP`	Adresse IP au format "xxx.xxx.xxx.xxx".
`retour`	Adresse IP sous forme d'un entier.

Entier signé ou non
Un simple echo du résultat fourni par ip2long peut conduire à l'affichage d'un entier négatif. Or celui-ci est théoriquement non signé. Il est donc préférable de faire appel à `printf("%u", ip2long($adresseIP))`.

long2ip()

Convertit une adresse IP d'entier au format "xxx.xxx.xxx.xxx".

Syntaxe	`int ip2long(string $adresseIP)`
`$adresseIP`	Adresse IP sous forme d'un entier.
`retour`	Adresse IP au format "xxx.xxx.xxx.xxx".

Dans certaines circonstances, comme par exemple pour déterminer si une adresse e-mail a des chances d'être valide, il peut être utile de déterminer si le nom de domaine indiqué existe, ou, plus précisément, vérifier si ce nom est connu du DNS.

DNS
DNS sont les initiales anglaises de "Domain Name Server", autrement dit "Serveur de nom de domaine". En deux mots, ce serveur contient les tables de correspondances qui permettent de retrouver une machine (ou son adresse IP) à partir de son nom. Vous trouverez plus d'informations sur le site Internet : ***http://www.nic.fr/guides/dns-intro***.

Pour tester la présence d'un nom de machine auprès du DNS vous ferez appel à `checkDNSRR()`.

checkDNSRR() (non disponible sous Windows)

Teste la présence d'un nom de machine ou d'une adresse IP auprès du DNS. Notez que dans la version 5 de PHP, cette fonction est baptisée `DNS_check_record()` (tout en assurant la compatibilité).

Syntaxe	`boolean checkDNSRR(string $nomMachine [, string $type])`
`$nomMachine`	Nom de la machine ou adresse IP.
`$type`	Précise le type d'entrée recherchée (par défaut MX) : `A` (Address = Adresse) : une simple adresse. `CNAME` (Cannonical Name = Nom canonique) : un alias. `MX` (Mail eXchanger = Distributeur de courrier) : machine permettant la réception de mails. `NS` (Name Server = Serveur de nom) : un serveur de nom. `PTR` (Pointer = Pointeur) : un renvoi sur une autre machine. `SOA` (Start Of Authority) : une zone. `ANY` : l'ensemble des options précédentes.
`retour`	`TRUE` si la machine est connue du DNS (pour le type précisé), `FALSE` sinon.

Voici donc une petite fonction permettant non pas de vérifier la validité d'une adresse e-mail, mais de débusquer certaines adresses manifestement non valides.

```
<?php
    function testEmail($email)
    {
        $domaine = strstr($email, '@');
        return checkdnsrr($domaine, 'MX');
    }
    $email = "damien@toutestfacile.com";
    if (testEmail($email)) {
        echo "Je ne peux pas assurer que cette adresse email est valide ".
             "mais elle n'est pas totalement farfelue";
    } else {
        echo "Pfuuu... C'est n'importe quoi cet email, ".
             "jamais je ne pourrai envoyer d'email à cette adresse";
    }
?>
```

Pour ce qui concerne les entrées MX du fichier de configuration du DNS, il est possible d'en savoir un peu plus grâce à la fonction `getMXRR()`.

getMXRR() (Non disponible sous Windows)

Retourne la liste des machines enregistrées auprès du DNS pour la gestion des mails. Notez que dans la version 5 de PHP, cette fonction est baptisée `DNS_get_mx()` (tout en assurant la compatibilité).

Syntaxe	`boolean getMXRR(string $nomMachine, array &$machines [, array &$poids])`
`$nomMachine`	Nom de la machine.
`$machines`	Référence sur une variable dans laquelle sera copié un tableau indexé contenant les noms des machines devant "router" les mails.

`$poids`	Référence sur une variable dans laquelle sera copié un tableau indexé contenant les poids associés aux machines. Les mails seront prioritairement "routés" par la machine de plus faible poids.
`retour`	`TRUE` si la machine est connue du DNS, `FALSE` sinon.

Protocoles et services

PHP propose également des fonctions permettant de récupérer des informations plus intimement liées au serveur.

Il est, par exemple, possible de connaître le port associé à un service ou, inversement, de retrouver le nom d'un service à partir de son numéro de port.

getServByName()

Retourne le port associé au service donné.

Syntaxe	`int getServByName(string $service, string $protocole)`
`$service`	Nom du service (ex. : "ftp", "http", ...)
`$protocole`	Nom du protocole ("tcp" ou "udp").
`retour`	Numéro de port, ou `FALSE` en cas d'échec.

getServByPort()

Retourne le nom du service associé au port donné.

Syntaxe	`int getServByPort(string $port, string $protocole)`
`$port`	Numéro du port.
`$protocole`	Nom du protocole ("tcp" ou "udp").
`retour`	Nom du service, ou `FALSE` en cas d'échec.

Ainsi, pour connaître le port associé au service FTP, ou le nom du service associé au port 80, l'on pourra utiliser le script suivant :

```
<?php
    echo getServByName("ftp", "tcp")."<br />";
    echo getServByPort(80, "tcp");
?>
```

ce qui retournera (probablement) :

```
21
http
```

Il est également possible de connaître le numéro associé à un protocole ou, inversement, de retrouver le nom d'un protocole à partir de son numéro.

getProtoByName()

Retourne le numéro associé à un nom de protocole (tel qu'on le trouve dans le fichier */etc/protocols* des systèmes UNIX/Linux).

Syntaxe	`int getProtoByName(string $nomProtocole)`
`$nomProtocole`	Nom du protocole.
`retour`	Numéro du protocole, ou `-1` s'il n'existe pas.

getProtoByNumber()

Retourne le nom associé à un numéro de protocole (tel qu'on le trouve dans le fichier */etc/protocols* des systèmes UNIX/Linux).

Syntaxe	`int getProtoByNumber(string $numeroProtocole)`
`$numeroProtocole`	Numéro du protocole.
`retour`	Nom du protocole, ou `FALSE` s'il n'existe pas.

Voici un exemple d'utilisation.

Le script suivant :

```
<?php
echo getProtoByName("tcp")."<br />";
echo getProtoByNumber(62)."<br />";
?>
```

pourra retourner :

```
6
cftp
```

16.3. Les sockets

Pour communiquer directement avec un service d'une machine donnée, vous serez peut-être amené à utiliser les sockets. Vous pouvez ainsi communiquer directement avec un serveur FTP, HTTP, NNTP (newsgroup), etc , et maîtriser plus finement les opérations que si vous utilisiez les commandes de plus haut niveau (plus couramment utilisées).

Le principe d'utilisation est simple et se déroule en quatre grandes étapes :

- Ouverture de la connexion ;
- Configuration de la connexion ;
- Lecture/écriture sur la socket ;
- Fermeture de la connexion.

Un exemple d'application est donné à la fin de ce sous-chapitre.

Ouverture de la connexion

L'ouverture d'une connexion se fait via la fonction `fSockOpen()`.

fSockOpen()

Ouvre une connexion sur une socket.

Syntaxe	`resource fSockOpen(string $serveur, int $port [, int &$codeErreur [, string &$msgErreur [, double $delaiExpiration]]])`
`$serveur`	Nom du serveur sur lequel doit se porter la connexion (éventuellement précédé de "udp://" pour une connexion UDP).
`$port`	Numéro du port sur lequel établir la connexion.
`$codeErreur`	Référence sur une variable dans laquelle sera copié le code d'erreur levé.
`$msgErreur`	Référence sur une variable dans laquelle sera copié le message d'erreur levé.
`$delaiExpiration`	Délai (en secondes) au-delà duquel la tentative de connexion doit être abandonnée (par défaut 60 secondes).
`retour`	Identifiant de connexion à la socket, ou `FALSE` en cas d'erreur.

La fonction `fSockOpen()` possède un équivalent permettant l'ouverture d'une connexion persistante (nous n'avons pas vérifié le caractère persistant et réutilisable de la connexion). Il s'agit de la fonction `pFSockOpen()`, qui possède exactement la même syntaxe que `fSockOpen()`.

Configuration de la connexion

Il est possible de jouer sur deux paramètres de connexion :

- Le mode de lecture bloquant ou non ;
- Le délai d'expiration (timeout) de la socket.

La lecture est dite en mode bloquant si la fonction de lecture doit attendre qu'un message lui soit adressé pour "rendre la main" au programme. Dans le cas contraire, elle est dite non bloquante, et la fonction de lecture retourne simplement le contenu de la mémoire tampon

(espace stockant les messages nouvellement reçus) qui pourra éventuellement être vide (si le message n'est pas encore arrivé).

Par défaut, en l'absence d'appel à la fonction `socket_set_blocking()`, le mode de lecture est bloquant.

socket_set_blocking()

Détermine si le mode de lecture de la socket doit être ou non bloquant.

Syntaxe	`boolean socket_set_blocking(resource $idSocket, boolean $modeLecture)`
`$idSocket`	Identifiant de la socket tel que retourné par `fsockopen()`.
`$modeLecture`	`TRUE` si le mode de lecture doit être bloquant, `FALSE` sinon.
`retour`	`FALSE` en cas d'échec, `TRUE` sinon.

La durée de vie de la connexion à la socket peut être limitée par la fonction `socket_set_timeout()`.

socket_set_timeout()

Détermine la durée de vie maximale de la socket.

Syntaxe	`boolean socket_set_timeout(resource $idSocket, int $secondes, int $microsecondes)`
`$idSocket`	Identifiant de la socket tel que retourné par `fsockopen()`.
`$secondes`	Partie "secondes" de la durée de vie de la socket exprimée en "secondes:microsecondes".
`$microsecondes`	Partie "microsecondes" de la durée de vie de la socket exprimée en "secondes:microsecondes".
`retour`	`FALSE` en cas d'échec, `TRUE` sinon.

Lecture/écriture sur la socket

La lecture et l'écriture sur la socket s'effectuent à l'aide des fonctions qui ont déjà été vues dans le chapitre relatif aux fichiers.

Vous pourrez ainsi utiliser `fGets($idSocket, $nbOctets);` pour lire jusqu'à `$nbOctets` octets sur la socket, `fPuts($idSocket, $chaine);` pour envoyer au serveur une chaîne de caractères via la socket, ou encore `fEOF($idSocket);` pour tester si la fin de fichier (fin d'émission) a été atteinte.

Fermeture de la connexion

Le fermeture de la connexion s'opère exactement de la même façon que celle d'un fichier, c'est-à-dire par la commande `fClose($idSocket)`.

Informations sur la connexion

Pour connaître l'état d'une socket, vous pouvez faire appel à la fonction `socket_get_status()`.

socket_get_status()

Retourne quelques informations sur l'état de la socket.

Syntaxe	`array socket_get_status(resource $idSocket)`
`$idSocket`	Identifiant de la socket tel que retourné par `fsockopen()`.
`retour`	Tableau associatif possédant les clés : "`time_out`" associée à une valeur de type booléen précisant si la durée de vie de la socket a expiré ou non. "`blocked`" associée à une valeur de type booléen précisant si la lecture sur la socket se fait en mode bloquant ou non. "`eof`" associée à une valeur de type booléen précisant si un indicateur de fin de fichier a été détecté ou non. "`unread_bytes`" associée à une valeur de type entier précisant le nombre d'octets actuellement dans la mémoire tampon de lecture de la socket.

Application

Pour mettre en place une communication via une socket, l'essentiel est de bien connaître le langage de communication avec le serveur. Si vous avez conçu votre propre serveur, vous savez certainement comment il fonctionne… autrement, il faut se référer aux spécifications.

Dans le cas d'un serveur HTTP, le principe est assez simple et a été évoqué dans le chapitre *En-têtes*.

Vous pouvez vous reporter à l 'annexe "Les en-têtes" pour plus de détails.

Pour récupérer un document via la méthode `GET`, il suffit de lui envoyer les instructions "`GET <nom du document> HTTP/<norme HTTP>`", et, dans le cas de la norme 1.1, il faut au minimum communiquer l'en-tête `Host`. Pour récupérer la page d'accueil d'un site, il faudra donc, par exemple, envoyer les instructions suivantes :

```
GET / HTTP/1.1
Host: localhost
```

ce qui nous donne le script suivant :

Listing 16.1 : socket_01.php

```
<?php

    // Exemple de récupération d'un document via HTTP
    // en utilisant directement les sockets.

    $serveur  = "www.gnu.org";
    $document = "/";

    echo "<b>Lecture de $serveur$document</b><br />";

    $idSocket = fSockOpen($serveur, 80, $codeErreur, $msgErreur);
    if (!$idSocket) {
        echo "La connexion via la socket a échouée.<br />";
        echo "Code d'erreur: $codeErreur<br />";
        echo "Message d'erreur: $msgErreur<br />";
        die();
    }

    // Configuration de la connexion
    // en mode bloquant
    // et avec un timeout de 5 minutes
    socket_set_blocking($idSocket, TRUE);
    socket_set_timeout($idSocket, 5, 0);


    // Envoi de données au serveur
    fputs($idSocket, "GET $document HTTP/1.1\r\n");
    fputs($idSocket, "Host: localhost\r\n");
    fputs($idSocket, "\r\n");                  // Marque la fin de l'en-tête

    // Lecture de la réponse
    while (!feof($idSocket)) {
        $donnees = fgets($idSocket, 512);
        echo "<xmp>$donnees</xmp>";            // Affichage du code source
    }
?>
```

qui fournira, par exemple, le résultat suivant (début de la réponse uniquement) :

```
Lecture de www.gnu.org/
HTTP/1.1 200 OK
Date: Tue, 28 May 2002 15:39:34 GMT
Server: Apache/1.3.24 (Unix) Debian GNU/Linux mod_python/2.7.8 Python/1.5.2
Last-Modified: Fri, 24 May 2002 17:51:38 GMT
ETag: "1088107-2e6e-3cee7daa"
Accept-Ranges: bytes
Content-Length: 11886
Content-Type: text/html
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
<HEAD>
<TITLE>GNU's Not Unix! - the GNU Project and the Free Software Foundation
(FSF)</title>
<META HTTP-EQUIV="Keywords"
 CONTENT="GNU, FSF, Free Software Foundation, Linux, Emacs, GCC, Unix,
        Free Software, Operating System, GNU Kernel, HURD, GNU HURD">
<META HTTP-EQUIV="Description"
 CONTENT="Since 1983, developing the free Unix-like operating system GNU,
```

Comme vous pouvez le constater, nous récupérons ainsi non seulement le code source HTML de la page, mais également l'en-tête retourné par le serveur (le début du code source se trouvant juste après la première ligne laissée vide).

L'utilisation des sockets permet ici de récupérer l'en-tête retourné par le serveur (et ainsi, par exemple, de vérifier le "Content-type"), mais aussi de préciser notre propre en-tête de requête (en utilisant par exemple le "User-agent" d'Internet Explorer ou de Mozilla, et en testant différentes valeurs pour "Accept-Language", etc.).

16.4. FTP

Les fonctions de la bibliothèque `FTP` permettent d'accéder en tant que client à un serveur FTP.

Installation

Sous Windows

Que ce soit avec l'archive du PHP Group ou avec EasyPHP, les fonctions FTP sont intégrées à PHP.

Sous Linux

Vous devrez recompiler PHP avec l'option `--enable-ftp` (avec les versions 3 de PHP, il s'agissait de l'option `--with-ftp`).

Vous pouvez vous reporter au chapitre "Prise en main" pour plus de détails sur la façon de compiler PHP.

Vérification

Pour vérifier que le support FTP est activé, appelez un script contenant `<?php phpinfo(); ?>`. Celui-ci doit alors laisser apparaître :

Figure 16.1 : *phpinfo() FTP*

Les fonctions de base

Connexion/déconnexion

La première étape consiste, comme toujours, à se connecter au serveur.

ftp_connect()

Permet de se connecter à un serveur FTP.

Syntaxe	`resource ftp_connect(string $serveur [, int $port [, int $delaiExpiration]])`
`$serveur`	Adresse du site FTP.
`$port`	Port du serveur FTP si celui-ci est différent du port usuel 21.
`$delaiExpiration`	Temps en secondes pour toute commande avant abandon ; par défaut cette valeur vaut 90 secondes.
`retour`	Un identifiant de connexion ou `FALSE` en cas d'erreur.

Une fois connecté, il faut s'identifier (sans quoi rien n'est possible).

ftp_login()

Identification auprès du serveur.

Syntaxe	`boolean ftp_login(resource $idConnexion, string $identifiant, string $motDePasse)`
`$idConnexion`	Identifiant de connexion obtenu par `ftp_connect()`.
`$identifiant`	Identifiant (login) de l'utilisateur.
`$motDePasse`	Mot de passe utilisateur.
`retour`	`TRUE` si la connexion a pu se faire, `FALSE` sinon.

La déconnexion s'effectuera, quant à elle, grâce à la fonction `ftp_close()`.

ftp_close()

Permet de clore la connexion FTP.

Syntaxe	`boolean ftp_close(resource $idConnexion)`
`$idConnexion`	Identifiant de connexion obtenu par `ftp_connect()`.
`retour`	`TRUE` en cas de succès.

`ftp_close()` possède un alias appelé `ftp_quit()`.

Déplacement dans l'arborescence

Une fois que l'on est identifié, il est possible d'effectuer toutes les opérations courantes en ligne, comme, par exemple, changer de répertoire :

ftp_chdir()

Permet de changer de répertoire.

Syntaxe	`boolean ftp_chdir(resource $idConnexion, string $repertoire)`
`$idConnexion`	Identifiant de connexion obtenu par `ftp_connect()`.
`$repertoire`	Répertoire de destination.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

De son côté, `ftp_cdup()` permet de monter d'un niveau dans la hiérarchie des répertoires.

ftp_cdup()

Change de répertoire pour monter d'un niveau.

Syntaxe	`boolean ftp_cdup(resource $idConnexion)`
`$idConnexion`	Identifiant de connexion obtenu par `ftp_connect()`.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

Liste du contenu d'un répertoire

Il est possible également de voir le contenu d'un répertoire :

ftp_rawlist()

Permet d'obtenir une liste détaillée des fichiers d'un répertoire (et éventuellement de ses sous-répertoires). Le résultat est identique à `ls -l`.

Syntaxe	`array ftp_rawlist(resource $idConnexion, string $repertoire [, boolean $recursif])`
`$idConnexion`	Identifiant de connexion obtenu par `ftp_connect()`.
`$repertoire`	Répertoire à lister.
`$recursif`	`TRUE` si le contenu des sous-répertoires doit également être retourné, `FALSE` (valeur par défaut) sinon. (Option disponible depuis PHP 4.3.0)
`retour`	Tableau indexé des différents fichiers du répertoire avec leur détail (droits, taille, date).

ftp_nlist()

Permet d'obtenir une liste des fichiers d'un répertoire. Le résultat est identique à `ls`.

Syntaxe	`array ftp_nlist(resource $idConnexion, string $repertoire)`
`$idConnexion`	Identifiant de connexion obtenu par `ftp_connect()`.
`$repertoire`	Répertoire à lister.
`retour`	Tableau des différents fichiers du répertoire.

Création, suppression, renommage

ftp_mkdir()

Création d'un nouveau répertoire.

Syntaxe	`boolean ftp_mkdir(resource $idConnexion, string $repertoire)`
`$idConnexion`	Identifiant de connexion obtenu par `ftp_connect()`.
`$repertoire`	Nom du répertoire à créer avec, éventuellement, son chemin.
`retour`	`TRUE` si le répertoire a pu être créé, `FALSE` sinon.

ftp_rmdir()

Permet d'effacer un répertoire vide.

Syntaxe	`boolean ftp_rmdir(resource $idConnexion, string $chemin)`
`$idConnexion`	Identifiant de connexion obtenu par `ftp_connect()`.
`$chemin`	Répertoire à effacer avec, éventuellement, son chemin.
`retour`	`TRUE` si le répertoire a été effacé, `FALSE` sinon.

ftp_delete()

Permet d'effacer un fichier.

Syntaxe	`boolean ftp_delete(resource $idConnexion, string $chemin)`
`$idConnexion`	Identifiant de connexion obtenu par `ftp_connect()`.
`$chemin`	Chemin du fichier à effacer avec, éventuellement, son chemin.
`retour`	`TRUE` si le fichier a été effacé, `FALSE` sinon.

ftp_rename()

Permet de renommer un fichier ou un répertoire.

Syntaxe	`boolean ftp_rename(resource $idConnexion, string $ancienNom, $nouveauNom)`
`$idConnexion`	Identifiant de connexion obtenu par `ftp_connect()`.
`$ancienNom`	Ancien nom du fichier ou répertoire.
`$ancienNom`	Nouveau nom du fichier ou répertoire.
`retour`	`TRUE` si le fichier ou répertoire a pu être renommé, `FALSE` sinon.

Transfert de fichiers

Les fonctions de transfert de fichiers sont également disponibles. Il est ainsi possible de copier un fichier du serveur FTP vers le serveur web (`ftp_get()`) ou le contraire (`ftp_put()`).

ftp_get()

Permet de récupérer un fichier d'un serveur FTP.

Syntaxe	`boolean ftp_get(resource $idConnexion, string $cheminFichierLocal, string $cheminFichierDistant, int $mode [, int $offset])`
`$idConnexion`	Identifiant de connexion obtenu par `ftp_connect()`.
`$cheminFichierLocal`	Chemin avec le nom du fichier où enregistrer le fichier.
`$cheminFichierDistant`	Chemin avec le nom du fichier à récupérer.
`$mode`	`FTP_ASCII` (pour les fichiers textes) ou `FTP_BINARY` (pour les fichiers binaires).
`$offset`	Position dans le fichier distant du premier octet à transférer. (Paramètre ajouté depuis PHP 4.3.0)
`retour`	`TRUE` si le fichier a pu être copié, `FALSE` sinon.

ftp_put()

Télécharge un fichier du serveur web vers le serveur FTP.

Syntaxe	`boolean ftp_put(resource $idConnexion, string $fichierDistant, string $fichierLocal, int $mode [, int $offset])`
`$idConnexion`	Identifiant de connexion obtenu par `ftp_connect()`.
`$fichierDistant`	Destination sur le serveur web.
`$fichierLocal`	Chemin sur le serveur web local.
`$mode`	`FTP_ASCII` (pour les fichiers textes) ou `FTP_BINARY` (pour les fichiers binaires).
`$offset`	Position dans le fichier local du premier octet à transférer. (Paramètre ajouté depuis PHP 4.3.0)
`retour`	`TRUE` si le fichier a pu être copié, `FALSE` sinon.

Les fonctions `ftp_get()` et `ftp_put()` possèdent des variantes permettant d'utiliser un pointeur vers un fichier. Cela permet de garder le fichier accessible en lecture ou écriture une fois copié.

ftp_fget()

Permet de télécharger un fichier d'un serveur FTP dans un fichier (ou plus généralement un stream) ouvert.

Syntaxe	`boolean ftp_fget(resource $idConnexion, resource $idFichier, string $fichierDistant, int $mode [, int $offset])`
`$idConnexion`	Identifiant de connexion obtenu par `ftp_connect()`.
`$idFichier`	Identifiant (obtenu par `fopen()`) du fichier à remplir.

`$fichierDistant`	Nom du fichier distant à récupérer.
`$mode`	`FTP_ASCII` (pour les fichiers textes) ou `FTP_BINARY` (pour les fichiers binaires).
`$offset`	Position dans le fichier distant du premier octet à transférer. (Paramètre ajouté depuis PHP 4.3.0)
`retour`	`TRUE` si le fichier a pu être récupéré, `FALSE` sinon.

ftp_fput()

Permet de déposer le contenu d'un fichier ouvert (ou plus généralement un stream) sur un serveur FTP.

Syntaxe	`boolean ftp_fput(resource $idConnexion, string $fichierDistant, resource $idFichier, int $mode [, int $offset])`
`$idConnexion`	Identifiant de connexion obtenu par `ftp_connect()`.
`$fichierDistant`	Nom qu'aura le fichier sur le serveur FTP.
`$idFichier`	Identifiant (obtenu par `fopen()`) du fichier à copier.
`$mode`	`FTP_ASCII` (pour les fichiers textes) ou `FTP_BINARY` (pour les fichiers binaires).
`$offset`	Position dans le fichier local du premier octet à transférer. (Paramètre ajouté depuis PHP 4.3.0)
`retour`	`TRUE` si le fichier a pu être mis sur le serveur FTP, `FALSE` sinon.

Exemple d'application

Les fonctions vues jusque-là sont les fonctions les plus couramment utilisées, et nous permettent de réaliser un script de client FTP.

Dans le script présenté, l'utilisateur pourra naviguer sur le compte FTP, ajouter des fichiers, en supprimer, en renommer, ajouter et supprimer des répertoires, uploader ou télécharger des fichiers.

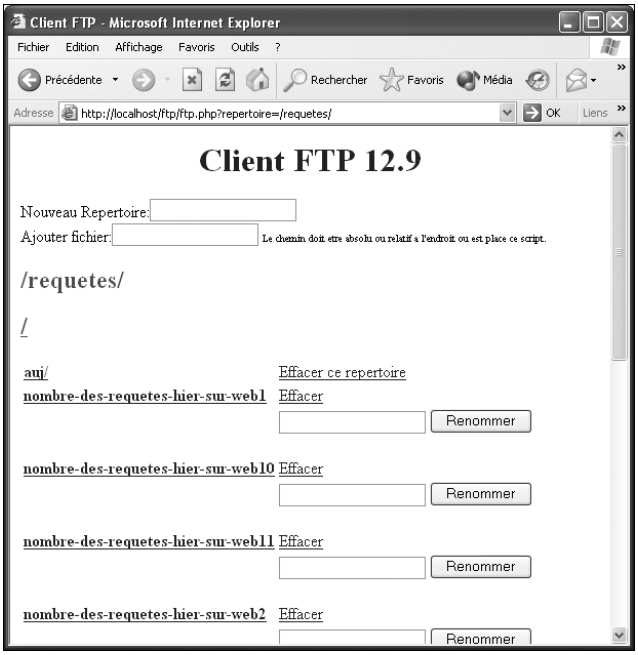


Figure 16.2 : *Client FTP en PHP*

Voici le script en question :

Listing 16.2 : ftp.php

```
<html>
  <head>
    <title>Client FTP</title>
    <link rel="stylesheet" type="text/css" href="style.css" />
  </head>
  <body>
    <center><font color="blue"><h1>Client FTP 12.9</h1></font></center>
    <font color="red">
<?php
    // On augmente le temps de vie du script à 100 secondes
    set_time_limit(100);
    // Parametres de connexion
    $serveur     = "ftp.monsite.com";
    $utilisateur = "utilisateur";
    $motdepasse  = "motdepasse";
    // Connexion au serveur FTP
    $connexion = ftp_connect($serveur)
        or die("Serveur FTP inexistant");
    // Identifiaction sur le serveur FTP
    ftp_login($connexion, $utilisateur, $motdepasse)
        or die("Utilisateur inconnu ou mauvais mot de passe");

    $repertoire = $_GET["repertoire"]!="" ? $_GET["repertoire"] : "/";
```

```
    // Verifie si "effacer" est passé en parametre
    if ($_GET["effacer"]!="") {
        effacer($_GET["effacer"], $repertoire);
    }
    // Verifie si "effacerrep" est passé en parametre
    if ($_GET["effacerrep"]!="") {
        effacerRep($_GET["effacerrep"], $repertoire);
    }
    // Verifie si "telecharger" est passé en parametre
    if ($_GET["telecharger"]!="") {
        telecharger($_GET["telecharger"], $repertoire);
    }
    // Verifie si "nouveaurep" est passé en parametre
    if ($_POST["nouveaurep"]!="") {
        nouveauRep($_POST["nouveaurep"], $repertoire);
    }
    // Verifie si "ajouterfichier" est passé en parametre
    if ($_POST["ajouterfichier"]!="") {
        ajouterFichier($_POST["ajouterfichier"], $repertoire);
    }
    // Verifie si "nouveaunom" est passe en parametre
    if ($_POST["nouveaunom"]!="") {
        nouveauNom($_POST["nouveaunom"], $_POST["fichier"], $repertoire);
    }
?>
    </font>
    <form method="post">
      Nouveau Repertoire:<input type="text" name="nouveaurep" /><br />
    <form>
    <form method="post">
      Ajouter fichier:<input type="text" name="ajouterfichier" />
      <font size="1">Le chemin doit etre absolou ou relatif a
                    l'endroit ou est place ce script.</font><br />
    <form>
<?php
    // Changement de repertoire
    ftp_chdir($connexion, $repertoire);
    // Affichage du contenu du repertoire
    listerRepertoire($repertoire);
    ftp_close($connexion);
?>
  </body>
</html>

<?php
function listerRepertoire($repertoire) {
    global $connexion;
    echo "<h2><font color=\"green\">".$repertoire."</font></h2>\n";
    // Affichage du lien vers le repertoire superieur
    if ($repertoire!="/") {
        echo "<a href=\"ftp1.php?repertoire=".
```

```
                substr(substr($repertoire, 0, -1), 0,
                       1+strrpos(substr($repertoire, 0, -1), "/"))."\">
                <h2><font color=\"green\">".
                substr(substr($repertoire, 0, -1), 0,
                             1+strrpos(substr($repertoire, 0, -1), "/")).
                "</font></h2></a>\n";
        }
        // Recuperation de la liste des fichiers
        $liste = ftp_rawlist($connexion, $repertoire);
        echo "<table>";
        foreach($liste as $fichier) {
          echo "<tr>";
            if (substr($fichier,0,1)=="d") {
                echo "<td>";
                echo "<a href=\"ftp1.php?repertoire=$repertoire".
                             substr($fichier,56)."/\">".
                     "<b>".substr($fichier,56)."/</b></a><br />\n";
                echo "</td><td>";
                echo "<a href=\"ftp1.php?repertoire=$repertoire&effacerrep=".
                      substr($fichier,56)."\">Effacer ce repertoire</a><br />\n";
            } else {
                echo "<td>";
                echo "<a href=\"ftp1.php?repertoire=$repertoire&telecharger=".
                      substr($fichier,56)."\"><b>".substr($fichier,56)."</b>\n";
                echo "</td><td>";
                echo "<a href=\"ftp1.php?repertoire=$repertoire&effacer=".
                      substr($fichier,56)."\">Effacer</a><br />\n";
                echo "</td>";
                echo "</tr><tr><td> </td><td>";
                echo "<form method=\"post\">".
                     "<input type=\"hidden\" name=\"fichier\"
                             value=\"$repertoire\"".substr($fichier,56)."\"/>
                      <input type=\"text\" name=\"nouveaunom\" />
                      <input type=\"submit\" value=\"Renommer\"/></form>";
                echo "</td>";
            }
          echo "</tr>";
        }
        echo "</table>";
    }
    // Efface un fichier
    function effacer($fichier, $repertoire) {
        global $connexion;
        if (@ftp_delete($connexion, $repertoire.$fichier))
            echo "Le fichier $fichier a ete supprime";
        else
            echo "Impossible d'effacer le fichier $fichier";
    }

    // Efface un repertoire vide
    function effacerRep($fichier, $repertoire) {
        global $connexion;
```

```
    if (@ftp_rmdir($connexion, $repertoire.$fichier))
        echo "Le repertoire $fichier a ete supprime";
    else
        echo "Impossible d'effacer le repertoire $fichier assurez vous "
             ."qu'il est vide";
}

// Telecharge un fichier depuis le site FTP
function telecharger($fichier, $repertoire) {
    global $connexion;
    if (@ftp_get($connexion, $fichier, $repertoire.$fichier, FTP_BINARY))
        echo "Le fichier $fichier devrait etre telecharge dans le repertoire".
             " ou se trouve ce script";
    else
        echo "Impossible d'ouvrir le fichier $fichier";
}

// Cree un repertoire
function nouveauRep($fichier, $repertoire) {
    global $connexion;
    if (@ftp_mkdir($connexion, $repertoire.$fichier))
        echo "Le nouveau repertoire $fichier a ete cree";
    else
        echo "Impossible de creer le repertoire $fichier";
}

// Renomme un fichier
function nouveauNom($nouveaunom, $fichier, $repertoire) {
    global $connexion;
    if (@ftp_rename($connexion, $fichier, $repertoire.$nouveaunom))
        echo "Le fichier $fichier a ete renomme";
    else
        echo "Impossible de renommer $fichier";
}

// Ajoute un fichier
function ajouterFichier($fichier, $repertoire) {
    global $connexion;
    $nomFichier = (strrchr($fichier, "/")) ?
                  $repertoire.substr(strrchr($fichier, "/"), 1) :
                  $repertoire.$fichier;
    if (@ftp_put($connexion, $nomFichier, $fichier, $FTP_BINARY))
        echo "Le fichier $fichier a ete ajoute";
    else
        echo "Impossible d'ajouter $fichier";
}

?>
```

Transfert de fichiers en mode asynchrone

Les fonctions `ftp_put()`, `ftp_get()`, `ftp_fput()` et `ftp_fget()` sont des fonctions synchrones. C'est à dire que lorsque le script PHP traite ces fonctions il ne fait plus rien d'autre en attendant le transfert complet des données. Depuis, PHP 4.3.0, ces fonctions ont leur équivalent en mode asynchrone. Il s'agit des fonctions `ftp_nb_put()`, `ftp_nb_get()`, `ftp_nb_fput()`, `ftp_nb_fget()`. Leur syntaxe est tout à fait identique si ce n'est qu'elles ne retournent pas un booléen mais un entier dont les valeurs possibles sont les mêmes que celles retournées par la fonction décrite ci-après. Une fois lancées, ces fonctions transfèrent bien les fichiers mais le script, lui, continue pour traiter les instructions suivantes. Pour savoir si la précédente opération est terminée vous devrez faire appel à la fonction `ftp_nb_continue()`.

ftp_nb_continue

Teste si la dernière opération FTP asynchrone lancée est terminée ou non.

Syntaxe :	`int ftp_nb_continue(resource $idConnexion)`
`$idConnexion`	Identifiant de connexion obtenu par `ftp_connect()`.
`retour`	`FTP_MOREDATA` si le transfert est en cours, `FTP_FINISHED` s'il est terminé ou `FTP_FAILED` s'il a échoué.

Autres fonctions

ftp_exec()

Permet d'exécuter une commande sur le serveur FTP si celui-ci l'autorise.

Syntaxe	`boolean ftp_exec(resource $idConnexion, string $commande)`
`$idConnexion`	Identifiant de connexion obtenu par `ftp_connect()`.
`$commande`	Commande à exécuter sur le serveur.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

ftp_site()

Envoie une commande à un serveur FTP de type SITE xxxx.

Syntaxe	`boolean ftp_site(resource $IdConnexion, string $commande)`
`$idConnexion`	Identifiant de connexion obtenu par `ftp_connect()`.
`$commande`	Commande à exécuter.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

Paramètres de connexion

ftp_get_option()

Retourne certains paramètres de la connexion FTP courante.

Syntaxe	`mixed ftp_get_option(resource $idConnexion, int $option)`
`$idConnexion`	Identifiant de connexion obtenu par `ftp_connect()`.
`option`	Option dont on veut connaître la valeur. Seul le temps avant abandon d'une commande est disponible ; la constante est `FTP_TIMEOUT_SEC`, la fonction retourne alors le temps en secondes.
`retour`	Dépend de l'option.

ftp_set_option()

Définit certains paramètres de la connexion FTP courante.

Syntaxe	`boolean ftp_set_option(resource $idConnexion, int $option, mixed $valeur)`
`$idConnexion`	Identifiant de connexion obtenu par `ftp_connect()`.
`option`	Option dont on veut définir la valeur. Seul le temps avant abandon d'une commande est disponible ; la constante est `FTP_TIMEOUT_SEC`.
`$valeur`	Valeur du paramètre à définir. Pour `FTP_TIMEOUT_SEC`, la valeur est à donner en secondes.
`retour`	`TRUE` si le paramètre a pu être modifié.

ftp_pasv()

Permet de passer en mode passif. Cela peut servir pour passer outre un firewall un peu exigeant.

Syntaxe	`boolean ftp_pasv(resource $idConnection, boolean $pasv)`
`$idConnexion`	Identifiant de connexion obtenu par `ftp_connect()`.
`$pasv`	`TRUE` pour activer le mode passif, `FALSE` sinon.
`retour`	`TRUE` si le changement de mode a pu s'effectuer, `FALSE` sinon.

Information sur les fichiers

ftp_mdtm()

Retourne la date de dernière modification d'un fichier.

Syntaxe :	`int ftp_mdtm(resource $idConnexion, string $fichierDistant)`
`$idConnexion`	Identifiant de connexion obtenu par `ftp_connect()`.
`$fichierDistant`	Fichier dont on veut connaître la date de dernière modification.
`retour`	La date de dernière modification au format *timestamp* d'UNIX.

ftp_size()

Retourne la taille d'un fichier.

Syntaxe	`int ftp_size(resource $idConnexion, string $fichierDistant)`
`$idConnexion`	Identifiant de connexion obtenu par `ftp_connect()`.
`$fichierDistant`	Fichier dont on veut connaître la taille.
`retour`	Taille du fichier en octets, `-1` si le fichier n'existe pas.

Informations sur le serveur

ftp_systype()

Retourne le type du système d'exploitation du serveur FTP (UNIX par exemple).

Syntaxe	`string ftp_systype(resource $idConnexion)`
`$idConnexion`	Identifiant de connexion obtenu par `ftp_connect()`.
`retour`	Le nom du système d'exploitation (attention, cela n'est absolument pas précis. Un serveur sous Linux renverra UNIX, mais cela permet de différencier les systèmes de fichiers de type UNIX et Windows.)

16.5. cURL (client URL Library)

La bibliothèque `cURL` a été développée afin de permettre le développement d'applications devant effectuer des opérations sur le réseau. Elle est disponible sur de nombreux systèmes : Linux, Windows, HPUnix, Solaris, Amiga, OS/2, MacOS X et d'autres encore.

`cURL` supporte différents protocoles parmi lesquels :

- FTP et FTPS ;
- HTTP et HTTPS ;

- GOPHER ;
- TELNET ;
- LDAP(v2) ;
- Le transfert de fichiers.

Vous pouvez retrouver tous les protocoles supportés par cURL sur cette page : **http://curl.haxx.se/docs/features.html**.

Elle gère les HTTP POST, HTTP PUT, l'upload de fichiers par FTP ou HTTP et passe les proxys. De plus, cURL sait manipuler les cookies et effectuer l'authentification HTTP.

Installation

Installation sous Linux

Si vous utilisez Linux, il est probable que vous ayez les librairies cURL installées sur votre machine. Si ce n'est pas le cas, vous pouvez toujours les télécharger sur le site web à l'adresse **http://curl.haxx.se/download.html** (ou utiliser la version disponible sur le CD-ROM).

Si vous souhaitez utiliser SSL, vous devrez également l'installer (ce qui est généralement déjà fait par défaut). OpenSSL est disponible à l'adresse **http://www.openssl.org/**. Nous supposerons ici qu'il a été installé sous */usr/local/ssl*.

Si vous avez récupéré les sources de cURL, commencez par décompresser l'archive en tapant :

```
# tar -zxvf curl-7.10.7.tar.gz
# cd curl 7.10.7
```

Ensuite, compilez cURL de la façon suivante :

```
# ./configure --disable-ipv6 --with-ssl=/usr/local/ssl
# make
# make install
```

Recompilez PHP en ajoutant l'option `--with-curl=/usr/local/lib` et puis relancez le serveur web.

Vérification

Appelez un simple script contenant `<?php phpinfo(); ?>`, vous devriez avoir un résultat similaire à :

curl

CURL support	enabled
CURL Information	libcurl 7.9.5 (OpenSSL 0.9.6b) (ipv6 enabled)

Figure 16.3 : *Vérifions que le module est activé*

Nous pouvons à présent nous concentrer sur la manipulation de ce module avec le langage PHP.

Utilisation

Initialiser une session cURL

Afin d'exploiter les différentes fonctions cURL, vous devez, dans un premier temps, créer une session avec l'instruction `curl_init()`.

curl_init()

Initialise une session cURL.

Syntaxe	`resource curl_init([string $adresse])`
`$adresse`	Adresse utilisée par les différentes fonctions cURL. Ce paramètre est optionnel, et peut être fixé ou modifié plus tard à l'aide de la fonction `curl_setopt()`.
`retour`	Ressource vers la session cURL initialisée.

Fermer une session cURL

La fermeture de la session est réalisée par l'appel à l'instruction `curl_close()`.

curl_close()

Ferme une session cURL.

Syntaxe	`void curl_close(resource $curlId)`
`$curlId`	Pointeur sur la ressource tel que retourné par `curl_init()`.

Préparer une requête

Avant d'effectuer votre requête, quelle qu'elle soit, vous devez spécifier certaines options à PHP. Pour cela, vous utiliserez la fonction `curl_setopt()`.

curl_setopt()

Permet de spécifier les options nécessaires à votre transfert cURL.

Syntaxe	`boolean curl_setopt (resource $curlId, string $option, mixed $valeur)`
`$curlId`	Pointeur sur la ressource tel que retourné par `curl_init()`.

`$option`	Option à modifier. Voyez le tableau ci-après pour connaître les différents paramètres configurables.
`$valeur`	Nouvelle valeur à donner à l'option.
`retour`	`TRUE` si la modification de l'option a été effectuée avec succès, `FALSE` dans le cas contraire.

Tableau 16.1 : Les différentes options configurables avec l'instruction curl_setopt()

Option	Description
`CURLOPT_COOKIEJAR`	Indique à PHP/cURL le fichier où doivent être stockés les différents cookies.

`CURLOPT_COOKIEFILE`	Indique le fichier contenant les différents cookies et leurs valeurs. Le fichier peut être dans même format que celui d'un en-tête HTTP :
	`Set-Cookie: cookie1=premier+cookie`
	`Set-Cookie: cookie2=deuxieme+cookie`
	`Set-Cookie: cookie3=troisieme+cookie`
	`Set-Cookie: cookie4=quatrieme+cookie`

`CURLOPT_CUSTOMREQUEST`	Permet d'indiquer une méthode particulière à envoyer au serveur : PUT, DELETE, etc.
`CURLOPT_FAILONERROR`	Indique à PHP/cURL de retourner les erreurs HTTP 300 et plus. Pour activer cette option, vous devez indiquer `TRUE` (ou `FALSE` sinon). Par défaut, cette option est desactivée.
`CURLOPT_FILE`	Spécifie le pointeur du fichier devant contenir les données de votre transfert. Ce fichier doit avoir été ouvert en écriture par l'instruction `fopen()`.
`CURLOPT_FOLLOWLOCATION`	Indique à PHP/cURL de suivre toutes les redirections HTTP envoyées par le serveur distant. Pour activer cette option vous devez indiquer `TRUE` (ou `FALSE` sinon). Par défaut, l'option est activée.
`CURLOPT_FTPAPPEND`	Indique à PHP/cURL de ne pas écraser les fichiers distants. À la place, le contenu du fichier est ajouté à la suite de celui existant. Pour activer cette option vous devez indiquer `TRUE` (ou `FALSE` sinon).
`CURLOPT_FTPLISTONLY`	Indique à PHP/cURL de n'effectuer qu'un listing des noms des fichiers sur un serveur FTP. Pour activer cette option, vous devez indiquer `TRUE` (ou `FALSE` sinon). Par défaut PHP retourne la liste complète des fichiers.
`CURLOPT_FTPPORT`	Spécifie au serveur distant l'adresse utilisée par PHP/cURL pour la connexion par FTP. Cette adresse est indiquée lors d'une connexion par FTP par la commande "PORT" envoyée au serveur.
`CURLOPT_HEADER`	Indique à PHP/cURL de retourner l'en-tête dans la réponse. Pour activer cette option, vous devez indiquer `TRUE` (ou `FALSE` sinon). Par défaut, PHP renvoie uniquement le corps du document.

Option	Description
CURLOPT_INFILE	Spécifie le pointeur de fichier contenant les données que vous expédiez lors d'un transfert. Ce fichier doit avoir été ouvert en lecture par l'instruction `fopen()`.
CURLOPT_INFILESIZE	Cette option sert à fixer la taille maximale des données à transmettre à un serveur distant.
CURLOPT_LOW_SPEED_LIMIT	Indique le nombre d'octets par secondes qui doivent circuler de façon à valider l'action de cURL. Si la vitesse indiquée n'est pas atteinte, PHP annulera l'exécution de l'action. Cette vitesse est calculée pendant une durée fixée par CURLOPT_LOW_SPEED_TIME.
CURLOPT_LOW_SPEED_TIME	Indique le temps en secondes qui est considéré pour vérifier le taux de transfert fixé par l'option CURLOPT_LOW_SPEED_LIMIT.
CURLOPT_MUTE	Indique à PHP/cURL de ne pas retourner les messages retournés par les différentes actions de cURL. Pour activer cette option, vous devez indiquer TRUE (ou FALSE sinon).
CURLOPT_NETRC	Indique à PHP/cURL d'utiliser le compte utilisateur et le mot de passe de l'utilisateur courant (tel que défini dans le fichier *./netrc* sous Linux) pour effectuer la connexion à distance. Pour activer cette option, vous devez indiquer TRUE (ou FALSE sinon).
CURLOPT_NOBODY	Indique à PHP/cURL de ne pas retourner le corps du document. Pour activer cette option, vous devez indiquer TRUE (ou FALSE sinon). Par défaut, l'option est desactivée.
CURLOPT_NOPROGRESS	Indique à PHP/cURL de retourner l'état des transferts avec le serveur distant. Pour activer cette option, vous devez indiquer TRUE (ou FALSE sinon). Par défaut, l'option est desactivée.
CURLOPT_POST	Indique à PHP/cURL de préparer une action de type HTTP POST (identique à l'envoie d'un formulaire HTML par méthode POST). Pour activer cette option, vous devez indiquer TRUE (ou FALSE sinon).
CURLOPT_POSTFIELDS	Indique à PHP/cURL les différentes données à passer lors d'un transfert HTTP par méthode POST. La chaîne doit être passée dans ce format : `var1=valeur1&var2=valeur2&var(n)=valeur(n)`.
CURLOPT_PROXYUSERPWD	Nom de l'utilisateur et mot de passe à utiliser lors de la connexion à un proxy HTTP. La chaîne indiquée est de la forme `nomUtilisateur:motPasse`.
CURLOPT_PUT	Indique à PHP/cURL de préparer une action HTTP de type PUT. Pour activer cette option, vous devez indiquer TRUE (ou FALSE sinon).
CURLOPT_RANGE	Permet de spécifier la plage de valeurs désirée. Vous devez la spécifier de la façon suivante : `val1-val2`. Vous pouvez préciser plusieurs plages différentes en les séparant par une virgule de la façon suivante : `val1-val2, val3-val4`.

Option	Description
CURLOPT_REFERER	Permet de spécifier l'en-tête REFERER envoyée lors d'une requête au serveur distant.
CURLOPT_RESUME_FROM	Indique à PHP/cURL le début du transfert. La valeur est fixée en octets.
CURLOPT_RETURNTRANSFER	Option permettant de récupérer le résultat dans une variable à l'exécution de la session : `$resultat = curl_exec($curlId)`. Pour activer cette option, vous devez indiquer TRUE (ou FALSE sinon). Par défaut, la fonction `curl_exec()` retourne un booléen.
CURLOPT_SSLCERT	Indique le fichier contenant le certificat SSL à passer au serveur. Ce certificat doit être au format PEM.
CURLOPT_SSLCERTPASSWD	Indique le mot de passe utilisé avec le certificat spécifié par l'option CURLOPT_SSLCERT.
CURLOPT_SSLVERSION	Indique la version de SSL utilisée pour les opérations avec cURL. La valeur peut être de 2 ou 3. Par défaut, PHP la déterminera automatiquement.
CURLOPT_STDERR	Spécifie le pointeur du fichier devant contenir les erreurs pouvant survenir lors d'un transfert. Ce fichier doit avoir été ouvert en écriture par l'instruction `fopen()`.
CURLOPT_TIMECONDITION	Indique à PHP/cURL comment utiliser l'option CURLOPT_TIMEVALUE. Les valeurs possibles sont TIMECOND_IFMODSINCE ou TIMECOND_ISUNMODSINCE. Par défaut, l'option est à TIMECOND_IFMODSINCE.
CURLOPT_TIMEOUT	Indique, en secondes, le temps maximum accordé à l'exécution d'une action par cURL.
CURLOPT_TIMEVALUE	Temps en secondes depuis le 1er janvier 1970.
CURLOPT_UPLOAD	Indique à PHP/cURL de préparer un transfert de fichier. Pour activer cette option, vous devez indiquer TRUE (ou FALSE sinon).
CURLOPT_URL	URL indiquant à PHP/cURL la page et le serveur distant. Cette URL désigne la page qui doit être récupérée. Cette option peut être fixée dès l'appel à l'instruction `curl_init()`.
CURLOPT_USERAGENT	Permet de spécifier l'en-tête USER-AGENT envoyé lors d'une requête au serveur distant.
CURLOPT_USERPWD	Nom de l'utilisateur et mot de passe de l'utilisateur qui doivent être utilisés lors de la connexion distante. La chaîne indiquée est de la forme `nomUtilisateur:motPasse`.
CURLOPT_VERBOSE	Indique à PHP/cURL d'afficher tous les événements. Pour activer cette option, vous devez indiquer TRUE (ou FALSE sinon).

Option	Description
CURLOPT_WRITEHEADER	Spécifie le pointeur du fichier devant contenir l'en-tête de sortie de votre transfert. Ce fichier doit avoir été ouvert en écriture par l'instruction `fopen()`.


RENVOI

De nombreux paramètres ont un lien étroit avec les en-têtes HTTP ; n'hésitez pas à consulter l'annexe "Les en-têtes" pour plus d'informations.

Récupérer les informations d'une option

Il est possible de récupérer les valeurs que contient une option en utilisant la fonction `curl_getinfo()`.

curl_getinfo()

Retourne la valeur d'une option.

Syntaxe	`string curl_getinfo(resource $curlId, int $option)`
`$curlId`	Pointeur sur la ressource tel que retourné par `curl_init()`.
`$option`	Option à récupérer.
`retour`	Chaîne de caractères indiquant la valeur de l'option qui y est associée.

Exécuter une session

Afin d'exécuter la session, il faut appeler la fonction `curl_exec()`.

curl_exec()

Exécute une session cURL.

Syntaxe	`mixed curl_exec(resource $curlId)`
`$curlId`	Pointeur sur la ressource tel que retourné par `curl_init()`.
`retour`	Par défaut, l'exécution de la fonction retourne `TRUE` si l'exécution de la session a été réalisée avec succès, `FALSE` dans le cas contraire. Suivant l'activation de certaines options, l'instruction peut retourner un entier ou une chaîne de caractères. Ainsi, l'option `CURLOPT_RETURNTRANSFER` indique à PHP de retourner le résultat plutôt que de l'afficher sur la sortie standard.

Gestion des erreurs

La gestion des erreurs est réalisée par l'appel à deux instructions : `curl_errno()` et `curl_error()`.

La fonction `curl_errno()` retourne le numéro du code de l'erreur.

curl_errno()

Permet de récupérer le code de la dernière erreur rencontrée pour une session cURL.

Syntaxe	`int curl_errno(resource $curlId)`
`$curlId`	Pointeur sur la ressource tel que retourné par `curl_init()`.
`retour`	Code de l'erreur cURL.

La fonction `curl_error()` permet de récupérer l'erreur dans une chaîne de caractères.

curl_error()

Retourne le dernier message d'erreur rencontré dans une session cURL.

Syntaxe	`string curl_error(resource $curlId)`
`$curlId`	Pointeur sur la ressource tel que retourné par `curl_init()`.
`retour`	Message d'erreur cURL.

Vérifier la version de cURL

L'instruction `curl_version()` permet de récupérer la version de votre librairie `cURL`.

curl_version()

Retourne la version de la bibliothèque `cURL`.

Syntaxe	`string curl_version(void)`
`retour`	Chaîne de caractères indiquant la version de la bibliothèque installée sur le système.

Exemples d'applications

À présent, nous allons voir différents exemples d'utilisation des fonctions cURL. Ce sont des exemples simples, qui n'ont d'autre but que de vous montrer les diverses possibilités de ce module.

Afficher le contenu d'une page web distante

Voici l'utilisation la plus simple des fonctions cURL. Le script va chercher la page sur un serveur distant et l'affiche sur le navigateur web.

Listing 16.3 : exemple1.php

```
<?php
$curlId = curl_init("http://www.linux.org/");
curl_exec($curlId);
curl_close($curlId);
?>
```

Récupérer le contenu d'une page dans un fichier

Cet exemple est une variation du précédent. Cette fois, nous spécifions simplement un fichier de sortie, ce qui permet de récupérer le code HTML de la page plutôt que de l'afficher directement.

Listing 16.4 : exemple2.php

```
<?php
// Ouverture d'un fichier en écriture
$fp = fopen("linux.html", "w");

// Création de la session cURL
$curlId = curl_init("http://www.linux.org/");

// Fichier de sortie
curl_setopt($curlId, CURLOPT_FILE, $fp);

// Exécution de la session
curl_exec($curlId);

// Fermeture de la session
curl_close($curlId);

// Fermeture du fichier
fclose($fp);

// On vérifie en affichant ensuite le fichier
readfile("linux.html");
?>
```

Envoyer des données par méthode POST

L'exemple suivant montre comment cURL peut envoyer des données par une méthode `POST`, cet envoi étant alors identique à celui depuis un formulaire HTML.

Listing 16.5 : exemple3.php

```
<?php
// Création de la session cURL
$curlId = curl_init("http://www.monserveur.com/formulaire.php");

// Le serveur demande une authentification
curl_setopt($curlId, CURLOPT_USERPWD, "utilisateur:mot2passe");

// Méthode POST
curl_setopt($curlId, CURLOPT_POST, TRUE);
$post = "nom=GUEDON".
        "&prenom=Laurent".
        "&mail=laurent@tild.com".
        "&action=valider";
curl_setopt($curlId, CURLOPT_POSTFIELDS, $post);

// Indiquons également différents cookies
$cookies = "cookie1=premier cookie;cookie2=deuxieme cookie";
curl_setopt($curlId, CURLOPT_COOKIE, $cookies);

// Execution de la session
curl_exec($curlId);

// Fermeture de la session
curl_close($curlId);
?>
```

Lister le contenu d'un répertoire FTP

Cet exemple liste les noms des fichiers se trouvant dans un répertoire. Le script récupère les différents fichiers dans une variable et les affiche ensuite en ajoutant la balise `<br>` entre les différents sauts de lignes.

Listing 16.6 : exemple4.php

```
<?php
// Création de la session cURL
$curlId = curl_init("ftp://www.monsite.org/repertoire/");

// Indiquons le login et le mot de passe de la connexion
curl_setopt($curlId, CURLOPT_USERPWD, "utilisateur:mot2passe");

// Indiquons de retourner la sortie dans une variable
curl_setopt($curlId, CURLOPT_RETURNTRANSFER, TRUE);

// On demande d'afficher uniquement les noms des fichiers
curl_setopt($curlId, CURLOPT_FTPLISTONLY, TRUE);

// Execution de la session
$retour = curl_exec($curlId);
```

```
// Affichons le résultat
echo nl2br($retour);

// Fermeture de la session
curl_close($curlId);
?>
```

Envoyer un fichier sur un serveur FTP

Cet exemple montre de façon simple comment envoyer un fichier sur un serveur FTP.

Listing 16.7 : exemple5.php

```
<?php
// Ouverture d'un fichier en lecture
$fp = fopen("monfichier.html", "r");

// Création de la session cURL
$curlId = curl_init("ftp://ftp.monserveur.com/html/monfichier.html");

// Indiquons le login et le mot de passe de la connexion
curl_setopt($curlId, CURLOPT_USERPWD, "utilisateur:mot2passe");

// Préparation d'un Upload de fichier
curl_setopt($curlId, CURLOPT_UPLOAD, TRUE);

// Taille du fichier à envoyer
curl_setopt($curlId, CURLOPT_INFILESIZE, filesize("monfichier.html"));

// Fichier à envoyer
curl_setopt($curlId, CURLOPT_INFILE, $fp);

// Indiquons l'adresse "PORT" du client
curl_setopt($curlId, CURLOPT_FTPPORT, "-");

// Execution de la session
if (!$donnee = curl_exec($curlId))
{
    die("Problème lors de l'envoie du fichier\n".
        curl_error($curlId));
} else {
    echo "Le fichier a été envoyé sur le serveur.";
}

// Execution de la session
curl_exec($curlId);

// Fermeture de la session
curl_close($curlId);
```

```
// Fermeture du fichier
fclose($fp);
?>
```

16.6. SOAP

SOAP (Simple Object Access Protocol) permet d'envoyer des messages XML à des serveurs, en utilisant HTTP comme moyen de communication.

Depuis peu, le célèbre moteur de recherche Google permet gratuitement de faire une requête sous forme XML via SOAP, et de récupérer le résultat de la requête. Ce sera le sujet de notre exemple.

Un message SOAP est généralement constitué d'un en-tête et d'un corps.

L'exemple que nous décrivons un peu plus loin dans ce chapitre donne lieu à la création du message SOAP suivant, que l'on appelle une enveloppe :

```
<?xml version="1.0" encoding="UTF-8"?>

<SOAP-ENV:Envelope  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:ns4="urn:GoogleSearch"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>

<ns4:doGoogleSearch>
<item xsi:type="xsd:string">M4liC3nceS3cr3tE</item>
<item xsi:type="xsd:string">mail site:www.toutestfacile.com</item>
<item xsi:type="xsd:int">0</item>
<item xsi:type="xsd:int">5</item>
<item xsi:type="xsd:boolean">true</item>
<item xsi:type="xsd:string"></item>
<item xsi:type="xsd:boolean">true</item>
<item xsi:type="xsd:string">lang_fr|lang_en</item>
<item xsi:type="xsd:string">latin1</item>
<item xsi:type="xsd:string">latin1</item></ns4:doGoogleSearch>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Dans l'exemple précédent, nous retrouvons, dans le corps de l'enveloppe, les informations pertinentes pour la fonction `doGoogleSearch` du serveur SOAP de Google.

Les fichiers WSDL (Web Service Definition Language) permettent de définir un service web ; ils sont écrits au format XML. À partir de ces fichiers, il est possible de savoir ce qu'attendent les services web comme paramètres. Voici, par exemple, un extrait du fichier WSL décrivant le service `doGoogleSearch` proposé par Google :

```
<?xml version="1.0"?>
```

```
<!-- WSDL description of the Google Web APIs.
     The Google Web APIs are in beta release. All interfaces are subject to
     change as we refine and extend our APIs. Please see the terms of use
     for more information. -->

<definitions name="urn:GoogleSearch"
             targetNamespace="urn:GoogleSearch"
             xmlns:typens="urn:GoogleSearch"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
             xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
             xmlns="http://schemas.xmlsoap.org/wsdl/">

  <!-- Types for search - result elements, directory categories -->

  <types>
    <xsd:schema xmlns="http://www.w3.org/2001/XMLSchema"
                targetNamespace="urn:GoogleSearch">

      <xsd:complexType name="GoogleSearchResult">
        <xsd:all>
          <xsd:element name="documentFiltering"
               type="xsd:boolean"/>
          <xsd:element name="searchComments"
               type="xsd:string"/>
          <xsd:element name="estimatedTotalResultsCount"
               type="xsd:int"/>
          <xsd:element name="estimateIsExact"
               type="xsd:boolean"/>
          <xsd:element name="resultElements"
               type="typens:ResultElementArray"/>
          <xsd:element name="searchQuery"
               type="xsd:string"/>
          <xsd:element name="startIndex"
               type="xsd:int"/>
          <xsd:element name="endIndex"
               type="xsd:int"/>
          <xsd:element name="searchTips"
               type="xsd:string"/>
          <xsd:element name="directoryCategories"
               type="typens:DirectoryCategoryArray"/>
          <xsd:element name="searchTime"
               type="xsd:double"/>
        </xsd:all>
      </xsd:complexType>

      <xsd:complexType name="ResultElement">
        <xsd:all>
          <xsd:element name="summary" type="xsd:string"/>
          <xsd:element name="URL" type="xsd:string"/>
          <xsd:element name="snippet" type="xsd:string"/>
```

```
        <xsd:element name="title" type="xsd:string"/>
        <xsd:element name="cachedSize" type="xsd:string"/>
        <xsd:element name="relatedInformationPresent" type="xsd:boolean"/>
        <xsd:element name="hostName" type="xsd:string"/>
        <xsd:element name="directoryCategory"
             type="typens:DirectoryCategory"/>
        <xsd:element name="directoryTitle" type="xsd:string"/>
      </xsd:all>
    </xsd:complexType>

    <xsd:complexType name="ResultElementArray">
      <xsd:complexContent>
        <xsd:restriction base="soapenc:Array">
           <xsd:attribute ref="soapenc:arrayType"
                wsdl:arrayType="typens:ResultElement[]"/>
        </xsd:restriction>
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="DirectoryCategoryArray">
      <xsd:complexContent>
        <xsd:restriction base="soapenc:Array">
           <xsd:attribute ref="soapenc:arrayType"
                wsdl:arrayType="typens:DirectoryCategory[]"/>
        </xsd:restriction>
      </xsd:complexContent>
    </xsd:complexType>

    <xsd:complexType name="DirectoryCategory">
      <xsd:all>
        <xsd:element name="fullViewableName" type="xsd:string"/>
        <xsd:element name="specialEncoding" type="xsd:string"/>
      </xsd:all>
    </xsd:complexType>

  </xsd:schema>
</types>


<!-- Messages for Google Web APIs - cached page, search, spelling. -->

<message name="doGoogleSearch">
  <part name="key"            type="xsd:string"/>
  <part name="q"              type="xsd:string"/>
  <part name="start"          type="xsd:int"/>
  <part name="maxResults"     type="xsd:int"/>
  <part name="filter"         type="xsd:boolean"/>
  <part name="restrict"       type="xsd:string"/>
  <part name="safeSearch"     type="xsd:boolean"/>
  <part name="lr"             type="xsd:string"/>
  <part name="ie"             type="xsd:string"/>
  <part name="oe"             type="xsd:string"/>
```

```
  </message>

  <message name="doGoogleSearchResponse">
    <part name="return"         type="typens:GoogleSearchResult"/>
  </message>

  <!-- Port for Google Web APIs, "GoogleSearch" -->
  <portType name="GoogleSearchPort">
    <operation name="doGoogleSearch">
      <input message="typens:doGoogleSearch"/>
      <output message="typens:doGoogleSearchResponse"/>
    </operation>
  </portType>

  <!-- Binding for Google Web APIs - RPC, SOAP over HTTP -->

  <binding name="GoogleSearchBinding" type="typens:GoogleSearchPort">
    <soap:binding style="rpc"
                  transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="doGoogleSearch">
      <soap:operation soapAction="urn:GoogleSearchAction"/>
      <input>
        <soap:body use="encoded"
                   namespace="urn:GoogleSearch"
                   encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </input>
      <output>
        <soap:body use="encoded"
                   namespace="urn:GoogleSearch"
                   encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
      </output>
    </operation>
  </binding>

  <!-- Endpoint for Google Web APIs -->
  <service name="GoogleSearchService">
    <port name="GoogleSearchPort" binding="typens:GoogleSearchBinding">
      <soap:address location="http://api.google.com/search/beta2"/>
    </port>
  </service>

</definitions>
```

Ce fichier, tel qu'il est écrit, se lit de bas en haut. Tout en bas, nous trouvons l'adresse du service web : `http://api.google.com/search/beta2` puis, au dessus, dans la balise `<binding>`, nous trouvons les différents services (ici, il n'y a que `doGoogleSearch`) où l'entrée et la sortie sont définies.

Dans la déclaration de la balise `<binding>`, le lien est fait vers le type `GoogleSearchPort`. On retrouve un peu plus haut ce type, qui définit les types d'entrées et de sorties. Les entrées seront donc de type `doGoogleSearch` et les sorties de type `doGoogleSearchResponse`.

Les types `doGoogleSearch` et `doGoogleSearchResponse` sont décrits plus haut dans le fichier.

Installation

SOAP fait partie du projet PEAR. Il dépend des paquetages PEAR suivants: *Mail_Mime*, *Net_URL*, *Net_Socket*, *HTTP_Request*, *Net_DIME*. Pour les installer vous pouvez, au choix, télécharger le fichier correspondant à l'adresse **http://pear.php.net/packages.html** et le dézipper dans un espace quelconque défini dans la variable *include_path* du fichier *php.ini*, ou bien vous pouvez utiliser les commandes PEAR. Notez toutefois, que pour les paquetages *SOAP* et *Net_DIME* qui n'ont pas encore le statut "stable" vous serez obligé de les télécharger avant de les installer avec les commandes Pear.

```
# pear upgrade Mail_Mime
# pear upgrade Net_URL
# pear upgrade Net_Socket
# pear upgrade HTTP_Request
# pear install Net_DIME.tar.gz
# pear install SOAP.tar.gz
```


REMARQUE

PHP 5

SOAP PEAR n'est (à l'heure de l'écriture de ces lignes) pas compatible avec PHP 5 (ceci à cause d'une mauvaise redéfinition de la méthode `__call` dans la classe `SOAP_Client`). Si vous vous sentez d'attaque vous pouvez tenter de le faire fonctionner sous PHP 5 en remplaçant la déclaration de la méthode `__call` par:

```
function __call($method, $args)
```

et en remplaçant `return TRUE;` par

```
return $this->call($method, $args);
```

Il n'est toutefois pas assuré que cela suffise pour un fonctionnement parfait.

Utiliser les classes PEAR

PEAR produit des classes pour les clients et serveurs SOAP, ainsi que des classes pour faire l'analyse lexicale d'un message SOAP, ou encore une classe qui permet de replacer le protocole standard HTTP (par la possibilité d'analyser le contenu d'un e-mail et de faire les appels contenus à l'intérieur de celui-ci).

Mais revenons à la classe a priori la plus utile : la classe `SOAP_Client` qui permet de faire appel à un serveur SOAP.

Le constructeur de cette classe nécessitera l'adresse du serveur SOAP.

SOAP_Client()

Constructeur d'objets SOAP_Client.

Syntaxe `SOAP_client SOAP_client(String $url [, boolean $wsdl [, string $nomPort]])`

`$url`	URL du serveur SOAP.
`$wsdl`	`TRUE` si l'URL est celle d'un fichier WSDL.
`$nomPort`	Nom du port SOAP utilisé par le client.
`retour`	Un objet de type SOAP_Client.

SOAP_Client->setEncoding()

Définit l'encodage des messages.

Syntaxe	`mixed setEncoding(string $encodage)`
`$encodage`	`'UTF-8'`, `'US_ASCII'` ou `'ISO-8859-1'`.
`retour`	`NULL`, ou une erreur de type `SOAP_Fault`.

SOAP_Client->addHeader()

Pour ajouter des en-têtes à l'enveloppe SOAP.

Syntaxe	`void addHeader(SOAP_Header $entete)`
`$entete`	En-tête à ajouter de type `SOAP_Header`.

SOAP_Client->call()

Permet de faire un appel au serveur SOAP.

Syntaxe	`array call(string $methode, array $parametres [, string $espaceNom, [string $actionSoap]])`
`$methode`	Nom de la méthode à appeler.
`$parametres`	Tableau des paramètres à passer.
`retour`	Tableau des résultats.

Interroger Google via PHP

Depuis début 2002, Google met à disposition un serveur SOAP permettant de faire des recherches dynamiquement. Attention, l'API fournie permettant d'interroger le moteur de recherche est susceptible d'être modifiée. Pire, ce service pourrait être supprimé du jour au lendemain.

Pré-requis

Pour pouvoir utiliser le serveur SOAP de Google, vous devrez récupérer un numéro de licence sur le site web de Google en créant un nouveau compte. Ce numéro de licence vous permettra de faire jusqu'à 1 000 requêtes par jour.

Nouveau compte
Pour créer un compte sur Google, l'adresse est :
https://www.google.com/accounts/NewAccount?continue=http://api.google.com/createkey &followup=http://api.google.com/createkey.

Vous serez également intéressé par le téléchargement de l'API, car c'est elle qui vous dira comment effectuer vos requêtes.

Télécharger l'API
Pour télécharger l'API de Google, l'adresse est :
http://www.google.com/apis/download.html.

Notre premier script sera très simple : il permet de faire une recherche avec des mots-clés intégrés dans le script. Les résultats seront affichés sans mise en page, mais juste à l'aide de la fonction `print_r()`.

Listing 16.8 : soapgoogle.php

```
<?php
    // On utilisera les classes de projet PEAR
    require_once "SOAP/Client.php";
    // On place ici le numero de licence obtenu sur le site de Google
    $numeroLicence = "MA_LICENCE, CETTE_CHAINE_DOIT_ETRE_REMPLACEE";
    // Les mots cles sont stockes dans le script ici.
    $motsCles = "Apprendre PHP facilement";
    // Creation du client SOAP et declaration du serveur SOAP.
    $soapClient = new SOAP_Client("http://api.google.com/search/beta2");
    // Les parametres a passer tels que decrits dans l'API de Google
    // (licence,
    // motscles,
    // indice du premier resultat a retourner,
    // nombre de resultats a retourner,
    // utilisation ou non du filtre cachant les resultats similaires,
    // restrictions a un pays ou domaine...,
    // filtrage des sites pour adultes,
    // Restriction sur la langue,
    // encodage d'entree,
    // encodage de sortie)

    $recherche = array($numeroLicence, $motsCles, 0, 1, TRUE, "",
                       TRUE, "", "", "");
    // Appel a la methode "doGoogleSearch" du serveur SOAP,
```

```
    $result = $soapClient->call("doGoogleSearch", $recherche,
              "urn:GoogleSearch");
    // Affichage du resultat
    print_r($result);
?>
```

Le résultat obtenu est le suivant :

```
stdClass Object
(
    [documentFiltering] => 1
    [estimatedTotalResultsCount] => 7080
    [directoryCategories] =>
    [searchTime] => 0.133035
    [resultElements] => Array
        (
            [item] => stdClass Object
                (
                    [cachedSize] => 22k
                    [hostName] =>
                    [snippet] => Un site pour bien apprendre le PHP et SQL , les
                    ✂ scripts commentés, le forum et les tutoriaux sont là pour
                    ✂ vous aider.
                    [directoryCategory] => stdClass Object
                        (
                            [specialEncoding] =>
                            [fullViewableName] => Top/World/Français/
                            ✂ Informatique/Programmation/Langages/PHP
                        )

                    [relatedInformationPresent] => 1
                    [directoryTitle] => <b>PHP</b> <b>Facile</b>
                    [summary] => <b>Apprendre</b> le <b>PHP</b>. On y trouve des
                    ✂ tutoriaux, des scripts commentés et un forum.
                    [URL] => http://www.phpfacile.com/
                    [title] => <b>PHP</b> <b>Facile</b> ! Le site pour
                    ✂ <b>apprendre</b> le <b>PHP</b> simplement et <b>...</b>
                )

        )

    [endIndex] => 1
    [searchTips] =>
    [searchComments] =>
    [startIndex] => 1
    [estimateIsExact] =>
    [searchQuery] => Apprendre PHP facile
)
```

Ce n'est pas plus difficile que ces quelques lignes…

À partir de là, il est facile d'élaborer un moteur de recherche personnel basé sur Google.

Voici notre interface graphique : elle permet d'entrer les mots-clés comme sur le site de Google, de préciser le nombre de résultats maximum à retourner, et le site sur lequel restreindre le recherche (il suffira de laisser le champ vide pour faire une recherche sur tous les sites).

Figure 16.4 : *Le formulaire rempli pour notre exemple*

Le script chargé de traiter la requête est le suivant :

Listing 16.9 : mp_perso.php

```
<html>
  <head>
    <title>Moteur de recherche personnel</title>
    <link rel="stylesheet" type="text/css" href="style.css" />
  </head>
  <body>
    <center><font color="blue">
      <h1>Resultat de la recherche</h1>
    </font></center>
<?php
    // On utilisera les classes de projet PEAR
    require_once "SOAP/Client.php";
    // On place ici le numero de licence obtenu sur le site de Google
    $numeroLicence = " MA_LICENCE, CETTE_CHAINE_DOIT_ETRE_REMPLACEE ";
    // Les mots cles sont stockes dans le script ici.
    $motsCles = $_POST["motscles"]." site:".$_POST["site"];
    $nbresultats = $_POST["nbresultats"];
    // Creation du client SOAP et declaration du serveur SOAP.
    $soapClient = new SOAP_Client("http://api.google.com/search/beta2");
    // Les parametres a passer tels que decrits dans l'API de Google
    // (licence,
    // motscles,
    // indice du premier resultat à retourner,
    // nombre de resultats à retourner,
    // utilisation ou non du filtre cachant les resultats similaires,
    // restrictions à un pays ou domaine...,
    // filtrage des sites pour adultes,
    // Restriction sur la langue,
    // encodage d'entrée,
    // encodage de sortie)

    $recherche = array($numeroLicence, $motsCles, 0,
```

```
                (int)$nbresultats, TRUE, "", TRUE,
                "lang_fr|lang_en", "latin1", "latin1");
    // Appel a la methode "doGoogleSearch" du serveur SOAP,
    $result = $soapClient->call("doGoogleSearch", $recherche,
              "urn:GoogleSearch");
    // Affichage du resultat
    echo "<font size=\"2\">";
    echo "<br />Temps de recherche:".$result->searchTime;
    echo "<br />Estimation du nombre de resultats:".
         $result->estimatedTotalResultsCount."<br />";
    echo "</font>";
    $i = 0;
    if ($result->resultElements) {
      foreach ($result->resultElements as $resultat) {
        $i++;
        echo "<br />$i -  <a href=\"".$resultat->URL."\">".
             $resultat->title."</a><br />";
        echo $resultat->snippet."<br />";
        echo "<font size=\"1\" color=\"#666666\">".
             $resultat->title."</font><br />";
        echo "<br />";
      }
    } else {
        echo "Pas de resultat";
    }
?>
  </body>
</html>
```

qui aura pour effet l'affichage suivant :

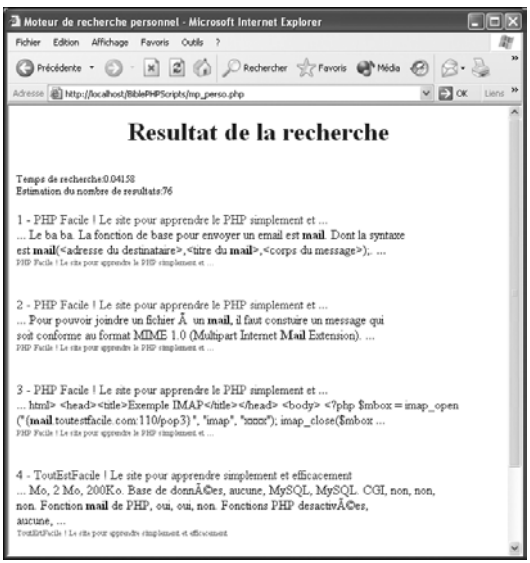


Figure 16.5 : *Résultat de la recherche*

Chapitre 17

Les processus et les identifiants

17.1. Exécution d'un programme

Les fonctions d'appel aux programmes extérieurs faisant partie intégrante de PHP, aucune bibliothèque n'est nécessaire pour cette partie.

Nous avons déjà vu l'utilisation des apostrophes inversées pour exécuter une commande, par exemple :

Listing 17.1 : processus_01.php

```
<?php
    echo 'dir c:/';
?>
```

pourra afficher :

```
Le volume dans le lecteur C n'a pas de nom.
 Le num,ro de s,rie du volume est A40D-542C

 R,pertoire de c:\program files\apache group\apache\htdocs\biblephpscripts\chap18

01/07/2002  22:25    <REP>          .
01/07/2002  22:25    <REP>          ..
01/07/2002  22:26                22 chap18-01.php
               1 fichier(s)              22 octets
               2 R,p(s)   1ÿ564ÿ647ÿ424 octets libres
```

Restrictions
L'utilisation des apostrophes inversées pour exécuter une commande n'est possible que si PHP n'est pas en "safe mode" (les hébergeurs sont souvent en safe mode) et si la fonction `shell_exec()` *n'a pas été désactivée.*

`shell_exec()` est identique à l'utilisation des apostrophes.

shell_exec()

Exécute et retourne le résultat de l'exécution.

Syntaxe	`string shell_exec(string $commande)`
`$commande`	Commande à exécuter.
`retour`	Le résultat de la commande.

Pour passer en argument une chaîne pouvant contenir des caractères à problèmes, la fonction `escapeshellarg()` permet de normaliser une chaîne de caractères destinée à être passée en argument.

escapeshellarg()

Permet de normaliser une chaîne de caractères destinée à être passée en argument.

Syntaxe	`string escapeshellarg(string $parametre)`
`$parametre`	Chaîne à normaliser.
`retour`	Chaîne normalisée.

Pour normaliser une commande et empêcher l'exécution d'une autre commande, la fonction `escapeshellcmd()` permet de s'assurer que la chaîne fournie sera normée.

escapeshellcmd()

Permet de normaliser une chaîne.

Syntaxe	`stringshellcmd(string $commande)`
`$commande`	Chaîne à normaliser.
`retour`	Chaîne normalisée.

Supposons, par exemple, que nous souhaitions lister le contenu d'un répertoire en nous basant sur une chaîne éventuellement saisie par un utilisateur via un formulaire. L'utilisateur est alors libre de taper ce qu'il veut et par exemple "`; rm -rf /;`" (sous Windows nous pourrions remplacer l'exemple par "`; format c:;`").

Listing 17.2 : processus_02.php

```
<?php
    $chaine="; rm -rf /;";
    echo "ls /tmp/".$chaine."<br />\n";
    echo "ls /tmp/".escapeshellcmd($chaine)."<br />\n";
?>
```

Voici le résultat obtenu :

```
ls /tmp/; rm -rf /;
ls /tmp/\; rm -rf /\;
```

Ici, nous n'avons affiché que les commandes. Dans le cas de la première commande, le serveur aurait perdu l'ensemble des fichiers s'il avait été exécuté, c'est pourquoi les hébergeurs sont peu enclins à laisser leurs clients utiliser les fonctions d'exécution de commandes...

Pour exécuter une commande, il est aussi possible d'utiliser la fonction `exec()` :

exec()

Permet d'exécuter un programme externe. En retour, on trouvera la dernière ligne du résultat de la commande.

Syntaxe	`string exec(string $commande[, array $sortie[, int $variableRetour]])`
`$commande`	Commande à exécuter.
`$sortie`	Tableau contenant toutes les lignes des sorties de la commande.
`$variableRetour`	Le code de retour sera inscrit dans cette variable.
`retour`	La dernière ligne de sortie.

Sécurité
Si vous permettez aux visiteurs de passer le moindre paramètre à cette fonction, n'oubliez pas d'appliquer les fonctions `escapeshellarg()` *et* `escapeshellcmd()` *décrites précédemment.*

passthru()

À la différence d'`exec()`, `passthru()` exécute la commande passée en paramètre et affiche la sortie du programme.

Syntaxe	`void passthru(string $commande [, int $variableRetour])`
`$commande`	Commande à exécuter.
`$variableRetour`	Le code de retour sera inscrit dans cette variable.

system()

Permet d'exécuter un programme externe et d'en afficher le résultat.

Syntaxe	`string system(string $commande [, int $variableRetour])`
`$commande`	Commande à exécuter.
`$variableRetour`	Le code de retour sera inscrit dans cette variable.
`retour`	La dernière ligne de la sortie de la commande.

17.2. POSIX

La bibliothèque `POSIX` n'est disponible que sous les systèmes de type UNIX (Linux, Mac OS X...), car elle fait appel à des notions de ce système d'exploitation.

Un fichier appartient nécessairement à un utilisateur et à un groupe. Cet utilisateur est caractérisé par deux paramètres. D'une part, il possède un nom d'utilisateur (le 'username' ou 'login name') et, d'autre part, tout utilisateur possède un identifiant unique de type entier appelé UID (User IDentifier). Même remarque pour le groupe qui a un nom (group name) et un identifiant unique nommé le GID (Group IDentifier).

À chaque fichier sont associés un UID et un GID. C'est le système qui, ensuite, fait le rapport entre ces identifiants et leur dénomination sur ce système. Le langage PHP possède une série de fonctions permettant de récupérer les UID et GID à partir des identifiants et vice versa. C'est le module POSIX qui fournit au langage cette interaction entre le système d'exploitation et l'interpréteur PHP.

Nom d'utilisateur et UID

Pour récupérer un identifiant à partir d'un UID, il faut passer par l'instruction `posix_getpwuid()`.

posix_getpwuid()

Retourne les informations sur l'utilisateur possédant l'UID donné en paramètre (telles que décrites dans le fichier */etc/passwd*).

Syntaxe	`array posix_getpwuid(int $uid)`
`$uid`	UID de l'utilisateur.
`retour`	Tableau associatif contenant les clés : "`name`", nom (login) de l'utilisateur. "`passwd`", mot de passe de l'utilisateur (crypté ou plus probablement `x` lorsqu'un fichier */etc/shadow* est utilisé). "`uid`", identifiant de l'utilisateur. "`gid`", groupe de l'utilisateur. "`gecos`", commentaire à propos de l'utilisateur. "`dir`", répertoire racine de l'utilisateur, en général on parle du répertoire home de cet utilisateur. "`shell`", système de commande de l'utilisateur. Il s'agit généralement de `bash`, `sh`, `csh` ou rien comme dans notre exemple.

```
<?php
$infoUtilisateur = posix_getpwuid(100);
while (list($key, $val) = each($infoUtilisateur)) {
    echo "$key = $val<br />";
}
?>
name = www
passwd = x
uid = 100
gid = 101
gecos = e-smith web server
```

```
dir = /home/e-smith
shell = /bin/false
```

Il est, bien entendu, possible de faire la même chose en ne connaissant, cette fois, que le nom de l'utilisateur. Ainsi, la fonction `posix_getpwnam()` retourne un tableau contenant les mêmes informations sur l'utilisateur que précédemment.

posix_getpwnam()

Retourne les informations sur un utilisateur en fonction de son nom (telles que décrites dans le fichier */etc/passwd*).

Syntaxe	`array posix_getpwnam(string $nomUtilisateur)`
`$nomUtilisateur`	Nom de l'utilisateur.
`retour`	Tableau associatif contenant les clés : "`name`", nom (login) de l'utilisateur. "`passwd`", mot de passe de l'utilisateur (crypté ou plus probablement '`x`' lorsqu'un fichier */etc/shadow* est utilisé). "`uid`", identifiant de l'utilisateur. "`gid`", groupe de l'utilisateur. "`gecos`", commentaire à propos de l'utilisateur. "`dir`", répertoire racine de l'utilisateur, en général on parle du répertoire home de cet utilisateur. "`shell`", système de commande de l'utilisateur. Il s'agit généralement de `bash`, `sh`, `csh` ou rien comme dans notre exemple.

Pour ne récupérer que le login de l'utilisateur propriétaire du processus courant, il suffit de faire appel à `posix_getlogin()`.

posix_getlogin()

Retourne le nom de login de la personne propriétaire du processus courant.

Syntaxe	`string posix_getlogin(void)`
`retour`	Le login en question.

Erreur
Dans nos essais nous avons eu comme retour :
*Warning: Cannot determine your login name. Something is really wrong here. in {chemin du script} on line {no Ligne}
.*
Ce qui signifie que le nom de login n'a pas pu être obtenu.

posix_geteuid()

Retourne le numéro d'identifiant du propriétaire effectif du processus courant.

Syntaxe	`int posix_geteuid(void)`
`retour`	Numéro d'utilisateur. Il est possible d'obtenir plus de détails sur ce groupe à l'aide de `posix_getpwuid()`.

posix_getuid()

Retourne le numéro d'identifiant du propriétaire réel du processus courant.

Syntaxe	`int posix_getuid(void)`
`retour`	Numéro d'utilisateur. Il est possible d'obtenir plus de détails sur ce groupe à l'aide de `posix_getpwuid()`.

posix_seteuid()

Permet de définir l'identifiant de l'utilisateur effectif du processus courant. Il faut avoir les privilèges adéquats pour ce type d'opération (il faut généralement être connecté en tant que `root`).

Syntaxe	`boolean posix_seteuid(int $identifiantUtilisateur)`
`$identifiant Utilisateur`	Identifiant de l'utilisateur auquel le processus courant doit être affecté.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

posix_setuid()

Permet de définir l'identifiant de l'utilisateur réel du processus courant. Il faut avoir les privilèges adéquats pour ce type d'opération (il faut généralement être connecté en tant que `root`).

Syntaxe	`boolean posix_setuid(int $identifiantUtilisateur)`
`$identifiant Utilisateur`	Identifiant de l'utilisateur auquel le processus courant doit être affecté.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

Nom de groupe et GID

Pour récupérer des informations sur un groupe, le développeur utilisera l'instruction `posix_getgrgid()`.

posix_getgrgid()

Retourne les informations sur le groupe possédant le GID donné en paramètre.

Syntaxe	`array posix_getpwuid(int $gid)`
`$gid`	GID du groupe.
`retour`	Tableau indexé et associatif. Les champs indexés contiennent les noms des différents utilisateurs appartenant au groupe, alors que les clés sont : "`name`", nom du groupe. "`gid`", identifiant du groupe. "`members`", nombre d'utilisateurs appartenant au groupe.

```
<?php
$infogroupe = posix_getgrgid(500);
while (list($key, $val) = each($infogroupe)) {
    echo "$key = $val<br />";
}
?>
```

```
name = shared
gid = 500
0 = public
1 = admin
2 = www
3 = laurent
4 = kangouroo
5 = jukebox
members = 6
```

Il est, bien entendu, possible de faire la même chose en ne connaissant, cette fois, que le nom du groupe. Ainsi, la fonction `posix_getgrnam()` retourne un tableau contenant les mêmes informations sur le groupe que précédemment.

posix_getgrnam()

Retourne les informations sur un groupe en fonction de son nom.

Syntaxe	`array posix_getgrnam(string $nomGroupe)`
`$nomGroupe`	Nom du groupe.

retour	Tableau indexé et associatif. Les champs indexés contiennent les noms des différents utilisateurs appartenant au groupe, alors que les clés sont : "name", nom du groupe. "gid", identifiant du groupe. "members", nombre d'utilisateurs appartenant au groupe.

```
<?php
echo "Information sur l'utilisateur Laurent<br />";
$infoUtilisateur = posix_getpwnam("laurent");
while (list($key, $val) = each($infoUtilisateur)) {
    echo "$key = $val<br />";
}
?>
<br />
<?php
echo "Information sur le groupe Laurent<br />";
$infogroupe = posix_getgrnam("laurent");
while (list($key, $val) = each($infogroupe)) {
    echo "$key = $val<br />";
}
?>
```

```
Information sur l'utilisateur Laurent
name = laurent
passwd = x
uid = 5001
gid = 5001
gecos = Laurent GUEDON
dir = /home/e-smith/files/users/laurent
shell = /bin/sshell

Information sur le groupe Laurent
name = laurent
gid = 5001
members = 0
```

posix_getegid()

Retourne le numéro du groupe effectif du processus courant.

Syntaxe	`int posix_getegid(void)`
retour	Numéro de groupe. Il est possible d'obtenir plus de détails sur ce groupe à l'aide de `posix_getgrgid()`.

posix_getgid()

Retourne le numéro du groupe réel du processus courant.

Syntaxe	`int posix_getgid(void)`
`retour`	Numéro de groupe. Il est possible d'obtenir plus de détails sur ce groupe à l'aide de `posix_getgrgid()`.

posix_getpgrp()

Retourne l'identifiant du groupe de processus courant.

Syntaxe	`int posix_getpgrp(void)`
`retour`	Un identifiant de groupe de processus.

posix_setegid()

Permet de définir l'identifiant du groupe effectif du processus courant. Il faut avoir les privilèges adéquats pour ce type d'opération (il faut généralement être connecté en tant que `root`).

Syntaxe	`boolean posix_setegid(int $identifiantGroupe)`
`$identifiantGroupe`	Identifiant du groupe auquel le processus courant doit être affecté.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

posix_setgid()

Permet de définir l'identifiant du groupe réel du processus courant. Il faut avoir les privilèges adéquats pour ce type d'opération (il faut généralement être connecté en tant que `root`).

Syntaxe	`boolean posix_setgid(int $identifiantGroupe)`
`$identifiantGroupe`	Identifiant du groupe auquel le processus courant doit être affecté.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

PID (identifiant de processus)

posix_getpid()

Retourne l'identifiant du processus courant.

Syntaxe	`int posix_getpid(void)`
`retour`	Un identifiant de processus.

posix_getppid()

Retourne l'identifiant du processus parent du processus courant.

Syntaxe	`int posix_getppid(void)`
`retour`	Un identifiant de processus.

posix_setpgid()

Permet de placer un processus dans un groupe de processus.

Syntaxe	`boolean posix_setpgid(int $IdentifiantProcessus, $identifiantGroupeProcessus)`
`$identifiant Processus`	Identifiant du processus à placer.
`$identifiant GroupeProcessus`	Identifiant du groupe dans lequel placer le processus.
`retour`	`TRUE` en cas de succès, `FALSE` sinon.

Chemins

posix_getcwd()

Retourne le nom du répertoire courant.

Syntaxe	`String posix_getcwd()`
`retour`	Le chemin complet du répertoire courant.

posix_ctermid()

Retourne le chemin complet du terminal.

Syntaxe	`string posix_ctermid(void)`
`retour`	Le chemin complet du terminal (ex. : */dev/tty*).

Ressources système

Lorsqu'une machine est partagée entre plusieurs utilisateurs, il peut être intéressant de limiter les ressources par utilisateur, afin, par exemple, qu'un des utilisateurs ne consomme pas 90 % des ressources tandis que les 10 % restants sont à partager entre une vingtaine d'autres utilisateurs.

La fonction `posix_getrlimit()` va nous permettre de connaître ces restrictions.

posix_getrlimit()

Permet de connaître les limites imposées de partage des ressources.

Syntaxe	`array posix_getrlimit()`
`retour`	Tableau associatif indiquant les différentes limites imposées.

```
<?php
    print_r(posix_getrlimit());
?>
```

```
Array
(
    [soft core] => 0
    [hard core] => unlimited
    [soft data] => unlimited
    [hard data] => unlimited
    [soft stack] => unlimited
    [hard stack] => unlimited
    [soft totalmem] => unlimited
    [hard totalmem] => unlimited
    [soft rss] => unlimited
    [hard rss] => unlimited
    [soft maxproc] => 2040
    [hard maxproc] => 2040
    [soft memlock] => unlimited
    [hard memlock] => unlimited
    [soft cpu] => unlimited
    [hard cpu] => unlimited
    [soft filesize] => unlimited
    [hard filesize] => unlimited
    [soft openfiles] => 1024
    [hard openfiles] => 1024
)
```

posix_times()

Récupère les différents temps du processus.

Syntaxe	`array posix_times()`
`retour`	Tableau associatif des différents temps : "`ticks`": nombre d'unités de temps écoulé depuis le dernier redémarrage de la machine. "`utime`": temps utilisateur du processus courant. "`stime`": temps système du processus courant. "`cutime`": temps utilisateur des processus enfants. "`cstime`": temps système des processus enfants.

```
<?php
    print_r(posix_times());
?>
```

```
Array
(
    [ticks] => 144168
    [utime] => 4
    [stime] => 2
    [cutime] => 0
    [cstime] => 0
)
```

Envoi d'un signal à un processus

posix_kill()

Envoie un signal à un processus dans le but de le terminer.

Syntaxe	`boolean posix_kill(int $identifiantProcessus, int $signal)`
`$identifiant Processus`	Identifiant du processus à terminer.
`$signal`	Numéro du signal à envoyer (`9` pour tuer le processus de manière brutale).
`retour`	`TRUE` si le signal a pu être envoyé, `FALSE` sinon.

Informations système

posix_uname()

Récupère des informations sur le système d'exploitation.

Syntaxe `array posix_uname()`

`retour` Tableau associatif des caractéristiques du système d'exploitation.

```
<?php
    print_r(posix_uname());
?>
```

```
Array
(
    [sysname] => Linux
    [nodename] => australia
    [release] => 2.2.19-7.0.8
    [version] => #1 Thu Jun 21 06:28:56 EDT 2001
    [machine] => i686
)
```


REMARQUE

domainname

Il est possible de trouver en plus, sur certaines machines, la clé `dommainname`*, qui correspond au nom de domaine du DNS. Ceci est une extension GNU qui ne fait pas partie de POSIX 1 ; c'est pourquoi nous ne retrouvons pas cette information ici.*

Chapitre 18

L'interopérabilité avec COM

18.1. COM

COM permet d'interagir avec la plupart des applications Microsoft courantes telles que Word, Excel, Powerpoint, Access, Outlook ou encore MSGraph.

Cette bibliothèque d'accès à la couche COM va nous permettre de faire réaliser par le serveur web toutes les opérations que nous pourrions faire manuellement en ouvrant ces mêmes applications.

Installation

La bibliothèque COM n'est disponible que sous Windows.

Que ce soit avec l'archive du PHP Group ou avec EasyPHP, vous n'aurez rien à faire de particulier pour en profiter, puisqu'elle fait partie du noyau PHP.

Toutefois, afin d'utiliser les constantes de COM, il faudra décommenter la ligne `com.autoregister_typelib = true` du fichier *php.ini*. De même, pour des appels distants, vous devrez mettre `com.allow_dcom` à la valeur `true`.

Utilisation

La manipulation des objets COM commence par l'instanciation d'un objet PHP par un simple appel du genre `new COM("nom application");`.

COM

Instancie un objet COM.

Syntaxe	`COM COM(string $application [, string $serveur [, int $encodage]])`
`$application`	Désignation de l'application.
`$serveur`	Nom ou adresse du serveur DCOM (par défaut `localhost`).

`$encodage`	Indique comment convertir les chaînes. Au choix :
	
	`CP_ACP.`
	`CP_MACCP.`
	`CP_OEMCP.`
	`CP_SYMBOL.`
	`CP_THREAD_ACP.`
	`CP_UTF7.`
	`CP_UTF8.`

`retour`	Objet COM.

Voici une série de désignations d'applications possibles :

- Access.Application ;
- Excel.Application ;
- Excel.Worksheet ;
- Excel.Chart ;
- MSGraph.Chart ;
- Outlook.Application ;
- Powerpoint.Application ;
- Powerpoint.Presentation ;
- Word.Application ;
- Word.Document.

Il existe ensuite une série de fonctions PHP permettant de lire ou modifier des attributs de ce composant, d'appeler des méthodes, etc. Mais tout cela peut se faire en utilisant simplement la notation objet de PHP.

Ainsi, pour modifier l'attribut `Visible` d'un objet Word et pour appeler la méthode `close()`, il suffira d'écrire :

```
<?php
$word = new COM("word.application")
        or die("Impossible de créer un objet Word");
$word->Visible = 1;
$word->Quit();
?>
```

La plus grande difficulté réside donc dans le fait de connaître les propriétés de ces objets. Il est probable que les adeptes de ces solutions propriétaires qui pratiquent déjà COM/DCOM s'y retrouveront sans mal. Pour les autres, sachez qu'il est facile de créer des scripts intéressants à moindre frais.

En fait, l'astuce consiste (c'est vrai pour Word, mais très probablement aussi pour les autres applications) à lancer l'enregistrement d'une macro (menu **Outils**/**Macro**/**Nouvelle macro**), à taper ce que vous souhaitez faire faire au serveur, à stopper l'enregistrement de la macro et, enfin, à consulter le résultat obtenu. Cette dernière opération pouvant se réaliser en allant dans le menu **Outils**/**Macro**/**Macros...**, en sélectionnant la macro nouvellement créée, et en sélectionnant l'option *modifier* qui fera apparaître le code de la macro.

Exemple 1 : impression d'une feuille de commande

Notre premier exemple permet, à partir d'un formulaire HTML, de générer automatiquement un document Word et d'en imprimer le contenu. L'exemple pris ici est celui de l'impression d'une feuille de commande. Il est alors possible d'imaginer, qu'à chaque nouvelle commande passée par un client, une feuille est automatiquement imprimée dans le bureau du magasinier, qui pourra alors se promener dans les rayons avec sa fiche. En Intranet, cela peut permettre à n'importe qui (dans un grand bureau) d'imprimer depuis son poste une étiquette au format

standard sur l'imprimante commune (dans ce cas, aucune installation spécifique n'est nécessaire sur le poste du client, hormis le navigateur qui est généralement présent par défaut ; en cas de modification du standard, seul le script du serveur sera à modifier et sans aucune intervention sur les postes clients).

La page d'appel est un simple formulaire dont voici une capture d'écran :

Figure 18.1 :
Formulaire de saisie

Cette page fait appel à une page, appelée *impression.php*, qui est chargée d'ouvrir un document Word, d'y écrire un contenu, d'imprimer puis de fermer le document.

Listing 18.1 : impression.php

```
<?php
$word = new COM("word.application")
        or die("Impossible de créer un objet Word");
// Juste à titre de test pour voir ce qu'il se passe
$word->Visible = 1;

$word->Documents->Add();

// Crée les tabulations
// Une à 8 cm du bord gauche de la feuille avec alignement au centre
// Une à 16 cm du bord gauche de la feuille avec alignement à droite
$word->Selection->ParagraphFormat->TabStops->Add(
    $word->CentimetersToPoints(8),
    wdAlignTabCenter,
    wdTabLeaderSpaces);
$word->Selection->ParagraphFormat->TabStops->Add(
    $word->CentimetersToPoints(16),
    wdAlignTabRight,
    wdTabLeaderSpaces);

// Saisie du texte gauche
$word->Selection->TypeText("Nom:");
$word->Selection->Font->Bold = wdToggle;
$word->Selection->TypeText(stripSlashes($_POST["nom"]));
```

```
// Passage au texte central par tabulation
$word->Selection->TypeText("\t");

// Saisie du texte central
$word->Selection->Font->Name  = "Times New Roman";
$word->Selection->Font->Size  = 20;
$word->Selection->Font->Bold  = TRUE;
$word->Selection->Font->Color = wdColorRed;
$word->Selection->TypeText("COMMANDE");

// Passage au texte droit par tabulation
$word->Selection->TypeText("\t");

// Saisie du texte droit
$word->Selection->Font->Size  = 12;
$word->Selection->Font->Bold = wdToggle;
$word->Selection->Font->Color = wdColorBlack;
$word->Selection->TypeText("Date:");
$word->Selection->Font->Bold = wdToggle;
$word->Selection->TypeText(strftime("%d/%m/%y"));
$word->Selection->Font->Bold = wdToggle;

// Passage au paragraphe suivant
$word->Selection->TypeParagraph();

// Saisie du paragraphe suivant en centré et italique
$word->Selection->ParagraphFormat->Alignment = wdAlignParagraphCenter;
$word->Selection->Font->Italic = TRUE;
$word->Selection->TypeText(stripSlashes($_POST["adresse"]));

// Passage au paragraphe suivant
$word->Selection->TypeParagraph();

// Saisie du paragraphe suivant en aligné à gauche
$word->Selection->ParagraphFormat->Alignment = wdAlignParagraphLeft;
$word->Selection->Font->Italic = FALSE;
$word->Selection->TypeText(stripSlashes($_POST["commande"]));

// Sélection de l'ensemble du texte
$word->Selection->WholeStory();

// Et encadrement en précisant les bords Haut, Bas, Gauche, Droit
$word->Selection->Borders[wdBorderTop]->LineStyle = wdLineStyleDouble;
$word->Selection->Borders[wdBorderTop]->LineWidth = wdLineWidth050pt;
$word->Selection->Borders[wdBorderTop]->Color = wdColorAutomatic;
$word->Selection->Borders[wdBorderBottom]->LineStyle = wdLineStyleDouble;
$word->Selection->Borders[wdBorderBottom]->LineWidth = wdLineWidth050pt;
$word->Selection->Borders[wdBorderBottom]->Color = wdColorAutomatic;
$word->Selection->Borders[wdBorderLeft]->LineStyle =wdLineStyleDouble;
$word->Selection->Borders[wdBorderLeft]->LineWidth = wdLineWidth050pt;
$word->Selection->Borders[wdBorderLeft]->Color = wdColorAutomatic;
$word->Selection->Borders[wdBorderRight]->LineStyle = wdLineStyleDouble;
```

```
$word->Selection->Borders[wdBorderRight]->LineWidth = wdLineWidth05Opt;
$word->Selection->Borders[wdBorderRight]->Color = wdColorAutomatic;

$word->Documents[1]->SaveAs("TestEtiquette.doc");

$word->Application->PrintOut();

$word->Quit();
?>
```


COM->Visible et WinNT/2000/XP

Si COM->Visible est mis à `vrai`, alors le document Word ne sera visible que par le propriétaire du document. Si le serveur web que vous avez installé est un service, alors le propriétaire est `SYSTEM`, (c'est-à-dire pas vous…) et vous ne verrez donc pas le document Word. Pour le voir, il faut lancer le serveur web "à la main". Pour Apache, il vous suffit d'arrêter le serveur et de lancer C:\progra~1\apache~1\apache\apache.exe.

Comment sommes-nous parvenus à ce résultat ? Simplement en générant le document, tout en enregistrant une macro.

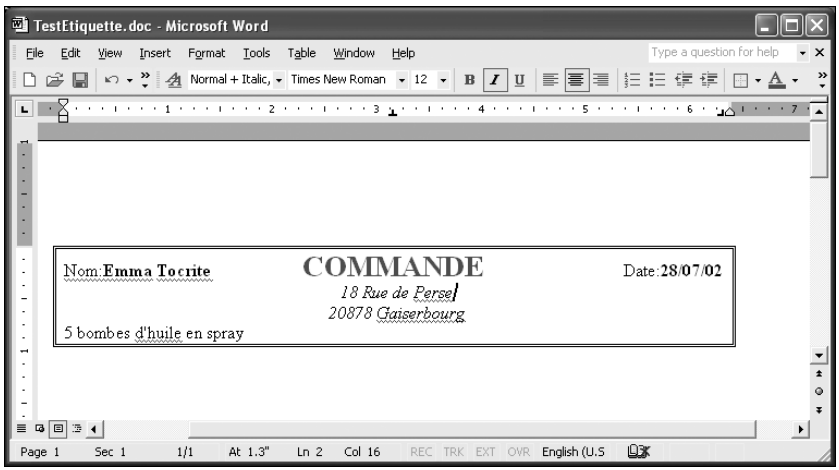


Figure 18.2 : *L'étiquette telle qu'elle sera imprimée par Word*

Ceci a eu pour effet de générer le code suivant pour la macro :

```
Sub Macro()
    Selection.ParagraphFormat.TabStops.ClearAll
    ActiveDocument.DefaultTabStop = CentimetersToPoints(1.25)
    Selection.ParagraphFormat.TabStops.Add Position:=CentimetersToPoints(8), _
        Alignment:=wdAlignTabCenter, Leader:=wdTabLeaderSpaces
    Selection.ParagraphFormat.TabStops.ClearAll
    ActiveDocument.DefaultTabStop = CentimetersToPoints(1.25)
    Selection.ParagraphFormat.TabStops.Add Position:=CentimetersToPoints(8),_
```

```
    Alignment:=wdAlignTabCenter, Leader:=wdTabLeaderSpaces
Selection.ParagraphFormat.TabStops.Add Position:=CentimetersToPoints(16),_
    Alignment:=wdAlignTabRight, Leader:=wdTabLeaderSpaces
Selection.TypeText Text:="Nom: "
Selection.Font.Bold = wdToggle
Selection.TypeText Text:="Le Nom" & vbTab
With Selection.Font
    .Name = "Times New Roman"
    .Size = 12
    .Bold = True
    .Italic = False
    .Underline = wdUnderlineNone
    .UnderlineColor = wdColorAutomatic
    .StrikeThrough = False
    .DoubleStrikeThrough = False
    .Outline = False
    .Emboss = False
    .Shadow = False
    .Hidden = False
    .SmallCaps = False
    .AllCaps = False
    .Color = wdColorRed
    .Engrave = False
    .Superscript = False
    .Subscript = False
    .Spacing = 0
    .Scaling = 100
    .Position = 0
    .Kerning = 0
    .Animation = wdAnimationNone
End With
Selection.TypeText Text:="COMMANDE" & vbTab
With Selection.Font
    .Name = "Times New Roman"
    .Size = 12
    .Bold = False
    .Italic = False
    .Underline = wdUnderlineNone
    .UnderlineColor = wdColorAutomatic
    .StrikeThrough = False
    .DoubleStrikeThrough = False
    .Outline = False
    .Emboss = False
    .Shadow = False
    .Hidden = False
    .SmallCaps = False
    .AllCaps = False
    .Color = wdColorAutomatic
    .Engrave = False
    .Superscript = False
    .Subscript = False
    .Spacing = 0
```

```
    .Scaling = 100
    .Position = 0
    .Kerning = 0
    .Animation = wdAnimationNone
End With
Selection.TypeText Text:="Date: "
Selection.Font.Bold = wdToggle
Selection.TypeText Text:="06/07/02"
Selection.TypeParagraph
Selection.ParagraphFormat.Alignment = wdAlignParagraphCenter
Selection.Font.Bold = wdToggle
Selection.Font.Italic = wdToggle
Selection.TypeText Text:="Adresse"
Selection.TypeParagraph
Selection.Font.Italic = wdToggle
Selection.TypeText Text:="Texte de la commande"
Selection.ParagraphFormat.Alignment = wdAlignParagraphLeft
Selection.WholeStory
With Selection.ParagraphFormat
    With .Borders(wdBorderLeft)
        .LineStyle = wdLineStyleDouble
        .LineWidth = wdLineWidth050pt
        .Color = wdColorAutomatic
    End With
    With .Borders(wdBorderRight)
        .LineStyle = wdLineStyleDouble
        .LineWidth = wdLineWidth050pt
        .Color = wdColorAutomatic
    End With
    With .Borders(wdBorderTop)
        .LineStyle = wdLineStyleDouble
        .LineWidth = wdLineWidth050pt
        .Color = wdColorAutomatic
    End With
    With .Borders(wdBorderBottom)
        .LineStyle = wdLineStyleDouble
        .LineWidth = wdLineWidth050pt
        .Color = wdColorAutomatic
    End With
    .Borders(wdBorderHorizontal).LineStyle = wdLineStyleNone
    With .Borders
        .DistanceFromTop = 1
        .DistanceFromLeft = 4
        .DistanceFromBottom = 1
        .DistanceFromRight = 4
        .Shadow = False
    End With
End With
With Options
    .DefaultBorderLineStyle = wdLineStyleDouble
    .DefaultBorderLineWidth = wdLineWidth050pt
    .DefaultBorderColor = wdColorAutomatic
```

```
    End With
End Sub
```

Une analyse rapide permet de comprendre qu'un objet Word possède un attribut `Selection`, qui possède des attributs et méthodes permettant de positionner une tabulation par appel à `Selection->ParagraphFormat->TabStops->Add()`, etc.

Il suffit, pour ainsi dire, de remplacer les points '.' par des flèches '->' pour passer du code de la macro à celui du script PHP. Puis de convertir les passages de paramètres "façon macro" en passage de paramètres PHP.

Il est facile d'identifier les constantes Word (elles commencent par `wd`) puis de deviner leur rôle.

Bref, avec un peu d'astuce, cela nous a permis d'obtenir le résultat escompté.

Exemple 2 : un Intranet facile !

L'idée de ce deuxième script est de réaliser un Intranet pour lequel on suppose que les personnes ayant du contenu à y insérer ne connaissent pas l'HTML, et se restreindront à utiliser Word et à enregistrer leurs documents au format Word.

Le script que nous allons créer permettra d'insérer un document Word au sein de deux autres définis comme l'en-tête et le pied de page du document.

Listing 18.2 : intranet.php

```
<?php
define("WDFORMATHTML", 8);

$word = new COM("word.application") or die("Unable to instanciate Word");
$word->Documents->Add();

// Charge le document d'entête
$word->Selection->InsertFile("C:\\entete.doc");
$word->Selection->InsertFile("C:\\secretaire.doc");
$word->Selection->InsertFile("C:\\pieddepage.doc");


$word->ActiveDocument->SaveAs($_SERVER["DOCUMENT_ROOT"].
                              "\\TestIntranet.html", WDFORMATHTML);
$word->Quit();
?>
```

Ce script ne fait qu'accoler les trois documents et enregistre le résultat en tant que fichier HTML à la racine du site web.

Voici le document Word d'en-tête :

Figure 18.3 : *entete.doc*

suivi du message de la secrétaire :

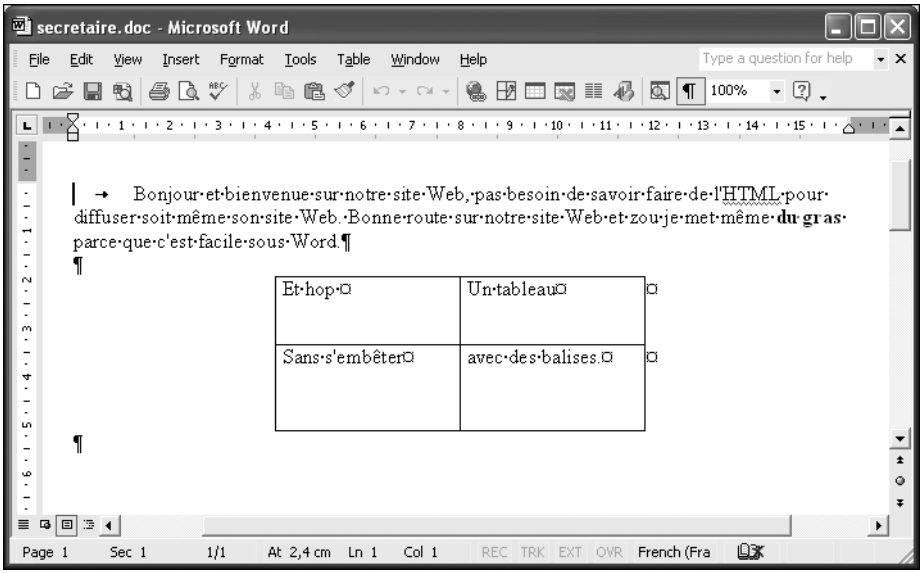


Figure 18.4 : *secretaire.doc*

puis du pied de page :

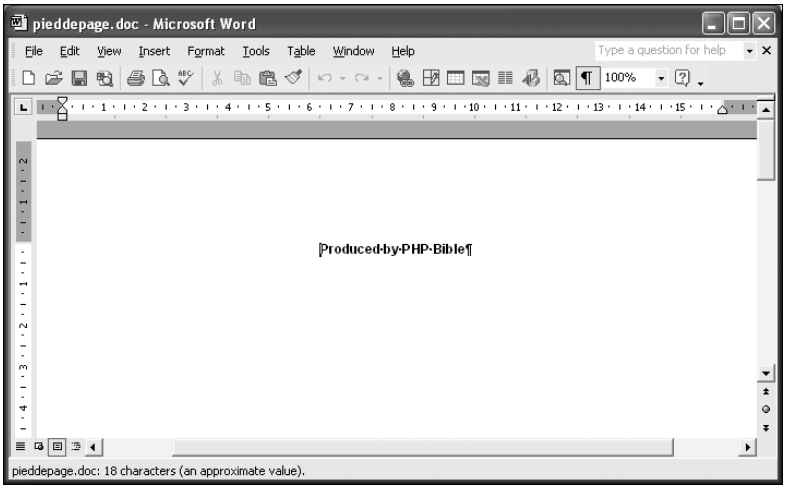


Figure 18.5 : *pieddepage.doc*

La page HTML produite par Word :

Figure 18.6 : *TestIntranet.html*

Comme vous pouvez le constater, l'en-tête est différent. Cela est dû à la façon de faire de Microsoft Word pour transformer un document Word en document HTML.

User de cette méthode peut provoquer de graves défaillances de serveur

Évitez tout de même d'user des objets COM comme les applications bureautiques, car ils sont très gourmands en ressources (chaque ouverture de l'application nécessite d'allouer beaucoup de mémoire et sollicite grandement le disque dur). Plus vous ouvrirez d'objets COM, plus la chute du serveur sera proche.

Chapitre 19

L'optimisation des temps de réponse

19.1. Introduction

Dès lors qu'un site devient populaire, et donc que le nombre quotidien des visiteurs est élevé, il convient de réduire au minimum le temps de réponse du serveur. Cela n'a pas pour seul bénéfice de réduire le temps d'attente de l'utilisateur, mais également de réduire la probabilité qu'une ou plusieurs requêtes arrivent alors que le serveur est déjà en train d'en traiter d'autres. Le nombre de requêtes que doit traiter simultanément un serveur est appelé "la charge". Plus le serveur sera "chargé", moins il aura de ressources disponibles, et plus il mettra de temps à répondre à une requête… jusqu'à ne plus pouvoir prendre en compte les nouvelles requêtes et être contraint de les refuser.

Ce temps de réponse dépend évidemment d'un grand nombre de paramètres :

- Le matériel (la quantité de mémoire, la vitesse des disques durs, etc.) ;
- Le système d'exploitation (sa capacité à réaliser plusieurs traitements en parallèle, sa vitesse d'exécution) ;
- Le serveur web (et sa capacité d'optimiser, donc de réduire le nombre d'opérations à réaliser).

Nous ne nous intéresserons ici qu'à l'aspect PHP (composante du serveur web).

L'exécution d'un script PHP se déroule en trois étapes principales :

1. Ouverture du fichier et analyse syntaxique du script (parsing) qui consiste à détecter les blocs d'instructions, les déclarations de fonctions, etc.
2. La compilation du code. Autrement dit, la transformation des éléments (écrits dans un langage compréhensible par l'homme) détectés dans l'étape 1 en éléments compréhensibles par la machine.
3. L'exécution du code (généralement pour générer un document HTML).

Nous allons vite nous rendre compte qu'il est possible d'intervenir à ces trois niveaux afin de réduire le temps de réponse du serveur.

La première des choses nécessaires pour que le temps d'exécution soit minimal est tout simplement que la compilation offre un code optimal (inutile de mettre deux instructions là où une seule suffit).

La seconde chose est d'éviter de refaire pour un client une opération qui vient d'être réalisée sur le même script pour un des clients précédents. Pour cela, il faut stocker le résultat des opérations 1, 2, et éventuellement 3, dans une mémoire appelée cache.

Ce dernier point doit être étudié avec attention. Il existe en effet deux cas de figure :

- C'est le résultat de la compilation (étapes 1 + 2) qui est mis en cache. Dans ce cas, le code sera ré-exécuté à chaque appel. Le document reste donc un document généré dynamiquement.
- C'est le résultat de l'exécution (étapes 1 + 2 + 3) qui est mis en cache (autrement dit le code HTML). Dans ce cas, le document n'est pas généré dynamiquement à chaque appel (c'est le code HTML qui est rappelé). Si, par exemple, ce document doit afficher l'heure, c'est l'heure à la date de mise en cache du document qui sera restituée lors des prochains appels.

La différence entre ces deux solutions est donc très importante. Alors que la première est applicable à n'importe quel type de script PHP, la seconde sera à réserver pour les scripts retournant un résultat évoluant peu dans le temps, ou, en tout cas, dont la mise à jour n'a pas besoin d'être immédiate (ceci n'est donc pas applicable à un forum et encore moins à un système de cotations en bourse).

Dans l'immédiat, nous ne nous intéresserons qu'aux solutions (que nous appellerons solutions "bas niveau") qui ne sont pas liées à une mise en cache du document généré.

Comme toujours avec les problèmes d'optimisation, il faut se méfier de la théorie. Les résultats peuvent être radicalement différents d'un cas de figure à l'autre. Afin d'avoir tout de même une idée de l'impact de ces différentes méthodes d'optimisation, nous avons réalisé des tests sur un mini-site local créé pour l'occasion. Ce site est assez fidèle à ce que peut contenir un véritable site web. Il contient donc des pages quasiment statiques (comme peuvent l'être les pages d'accueil) ainsi que des pages totalement dynamiques (comme peuvent l'être les pages de moteurs de recherche ou de forums). Pour mettre en avant certains cas de figures, nous avons tout de même rajouté des pages considérées comme non représentatives d'un site web (comme des pages faisant appel à du calcul scientifique).

19.2. Environnement de test des solutions "bas niveau" (modules PHP)

Configuration matérielle

Les tests ont été réalisés à partir d'un poste client basé sur un Intel Pentium II (Deschutes) 400 MHz équipé de 256 Mo de RAM et du système d'exploitation Linux (Mandrake 8.1) relié au serveur par une liaison BNC.

Le serveur, quant à lui, est équipé de la manière suivante :

- Processeur : AMD K6-II 350 MHz ;
- Mémoire : 192 Mo (+136 Mo virtuelle) ;
- Disque dur : IBM-DTTA-351010 10 Go (IDE) (11 Mo/s) ;
- Systèmes d'exploitation : Linux (Mandrake 9.1) ;
- Serveur : Apache 1.3.28 avec PHP 4.3.0 ;
- Base de données : MySQL 4.

Pourquoi PHP 4.3.0 ?
C'est la version 4.3.0 et non une version supérieure qui a été retenue parce qu'à la date où les tests ont été réalisés, il s'agissait de la seule version officiellement compatible avec l'ensemble des solutions d'optimisation présentées ici. Ceci dit, des tests complémentaires ont montré qu'il y a très peu de différences entre les résultats obtenus avec PHP 4.3.0 et PHP 4.3.2 même si celle-ci est en faveur de PHP 4.3.2.


REMARQUE

Et PHP 5 ?
À ce jour quasiment aucune des solutions d'optimisation ne supporte officiellement (du moins) PHP 5.

Pages testées

Les tests ont porté sur trois pages principales (dont les sources sont disponibles sur le CD-ROM) :

- Une quasi-statique ;
- Une dynamique ;
- Une mathématique.

La page quasi-statique

La page quasi-statique ne possède aucun élément variable. Tous les éléments de la page sont figés : aucun n'est dépendant de l'heure, de l'utilisateur ou du contenu d'une base de données. Ce peut-être, par exemple, une page d'accueil.

Bien que cette page aurait pu être écrite directement en HTML, sa constitution tire toutefois parti des avantages qu'offre PHP, puisque chaque élément de la page (en-tête, menu gauche, menu droit, pied de page) est décrit dans un fichier qui lui est propre. Le tout est assemblé par de simples "`include`". Les menus, quant à eux, sont décrits dans des classes (ce qui offre une grande facilité de mise à jour).

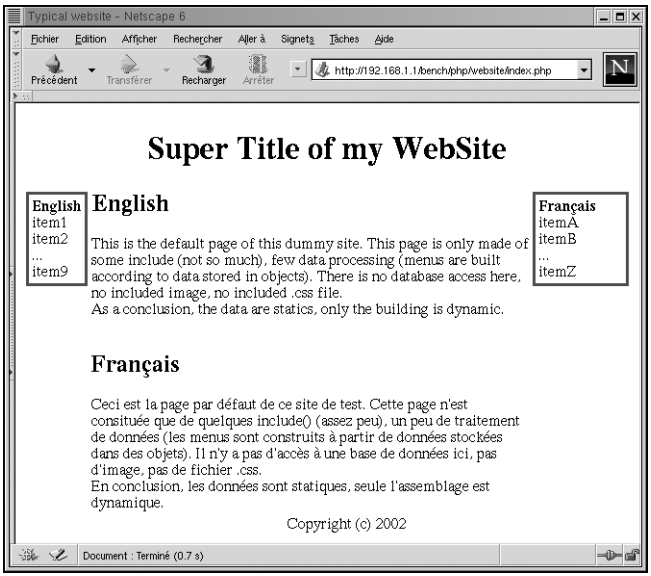


Figure 19.1 : *La page quasi-statique*

Cette page fait 1 569 octets.

La page dynamique

La page dynamique est, quant à elle, totalement dynamique et est susceptible de changer à chaque appel, puisque son contenu est issu d'une base de données pouvant être alimentée à tout moment. C'est le cas, par exemple, d'une page de résultats d'un moteur de recherche ou d'un forum.

Nous utiliserons ici le site de petites annonces développé au cours du chapitre *Les bases de données*.

La quantité de données dans la base est limitée à 33 enregistrements, afin que la mesure ne porte pas trop sur la capacité de réponse de la base de données.

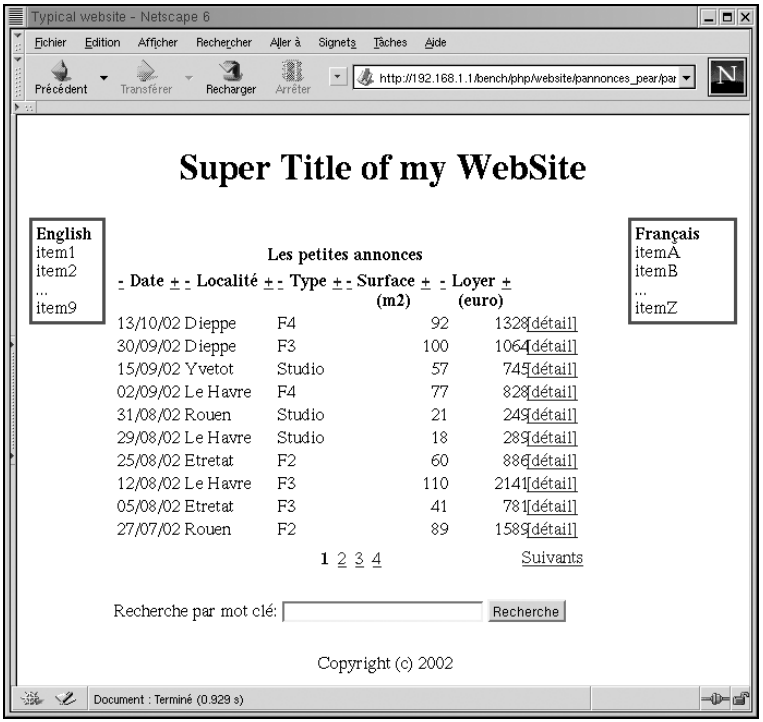


Figure 19.2 : *La page dynamique*

Cette page fait 8 850 octets.

La page mathématique

La page mathématique a pour objectif de réaliser un grand nombre d'opérations et de boucles, afin de véritablement solliciter le langage PHP lui-même. En l'occurrence, nous avons choisi de réaliser une fonction exponentielle qui effectue en boucle un grand nombre d'additions et de multiplications.

Ce cas n'est pas véritablement représentatif d'un site Internet.

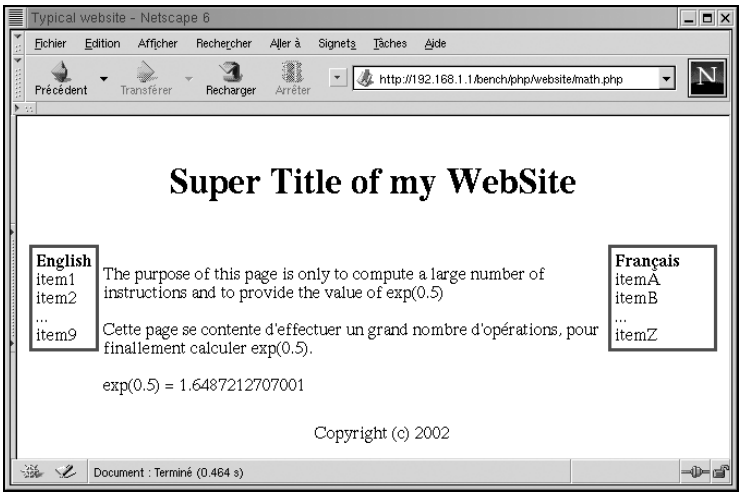


Figure 19.3 : *La page mathématique*

Cette page fait 1 082 octets.

Instrument de mesure

Les mesures ont été effectuées uniquement depuis le poste client (c'est donc le cycle complet requête-réponse qui a été mesuré) grâce à la commande `ab` fournie avec la version UNIX d'Apache.

Cette commande permet (entre autres) de préciser le nombre de requêtes à exécuter ainsi que le nombre de requêtes concurrentes, et retourne (entre autres) le temps global d'exécution et le nombre de requêtes qui n'ont pu être satisfaites.

Dans chacun des cas étudiés, nous avons systématiquement demandé l'exécution de 500 requêtes, et c'est donc toujours sur le temps nécessaire à la réalisation de ces 500 requêtes que nous avons établi la comparaison.

Les tests ont été déclinés avec les niveaux de concurrence suivants : 1 (les tests sont donc exécutés les uns après les autres), 3, 5, 10, 50 et 100.

Chaque test a été renouvelé plusieurs fois (au moins dix fois lors d'une session) et, afin d'écarter tout doute, la plupart des sessions de test ont été renouvelées deux fois à des dates différentes (avec redémarrage du système). Il était ainsi possible de détecter une éventuelle perte de performance liée à une forte occupation mémoire ou CPU par un programme externe lors du déroulement d'une des sessions.

Présentation des mesures

Les résultats des mesures (disponibles sur le CD-ROM) sont présentés sur un graphe accompagné d'un tableau donnant les valeurs numériques.

Chaque mesure est représentée par un point avec, pour ordonnée, la charge (le nombre d'accès concurrents) et, pour abscisse, le temps de réponse (temps total d'exécution des 500 requêtes, en secondes).

Pour chaque niveau de charge, le temps moyen d'exécution a été calculé afin de tracer la courbe d'évolution du temps d'exécution en fonction de la charge.

Le graphe intègre également (en arrière-plan et dans une autre échelle) le nombre moyen de requêtes en échec.

Notez que nous avons choisi une échelle logarithmique pour l'axe des abscisses (la charge).

Graphe
Les graphiques ont été réalisés grâce à la bibliothèque `JpGraph` *décrite en annexe de ce livre.*

19.3. En l'absence de solution d'optimisation

Mesures

Page quasi-statique

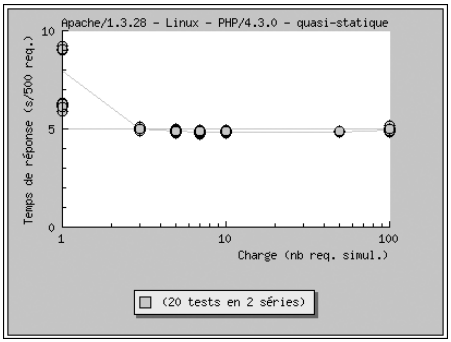


Figure 19.4 : *Temps de réponse d'une page quasi-statique sans optimisation*

Tableau 19.1 : Temps de réponse sans optimisation d'une page quasi-statique

Nombre de requêtes simultanées	Temps total d'exécution (en secondes)	Temps total d'exécution minimum (en secondes)	Temps total d'exécution maximum (en secondes)	Nombre de requêtes en échec
1	7.96	5.87	15.23	0
3	4.97	4.87	5.10	0
5	4.88	4.78	4.98	0
7	4.84	4.75	4.9	0

Nombre de requêtes simultanées	Temps total d'exécution (en secondes)	Temps total d'exécution minimum (en secondes)	Temps total d'exécution maximum (en secondes)	Nombre de requêtes en échec
10	4.85	4.78	4.9	0
50	4.86	4.81	4.91	0
100	4.92	4.85	5.17	0

Nous observons un délai d'exécution des 500 requêtes bien plus long (+ 64 %) lorsque celles-ci sont émises les unes à la suite des autres (en comparaison avec plusieurs appels simultanés). Ceci n'a rien d'étonnant, et vous comprendrez aisément qu'un système conçu pour traiter plusieurs tâches en même temps ne pourra optimiser son travail que si les demandes ne lui arrivent pas les unes après les autres.

Nous constatons également que le serveur ne souffre pas de la charge qui lui est imposée. Il traite quasiment aussi rapidement les requêtes qu'il y en ait 3 ou 100 simultanément (seule une très légère augmentation du temps d'exécution est observable à partir de 50 requêtes simultanées). Et aucune requête ne tombe en erreur (elles sont toutes honorées par le serveur).

Les résultats obtenus sont d'une très bonne stabilité. Hormis le cas où les requêtes sont émises les unes après les autres, les résultats obtenus d'un test à l'autre sont similaires (avec des variations maximales de plus ou moins 2 %, comme le montrent les temps maximum et minimum relevés).

Comparatif avec les tests des éditions précédentes

Les tests réalisés dans les mêmes conditions mais avec Mandrake 8.1, Apache 1.3.24, PHP 4.2.1 et MySQL 3 (sur le serveur) donnaient des résultats moins bons: de l'ordre de 5,8 secondes contre 4,8 ici. Difficile de déterminer quel est l'élément (PHP?) qui a permis ce gain en performance de 20% d'une année sur l'autre mais il est pourtant bel et bien réel.

Page dynamique

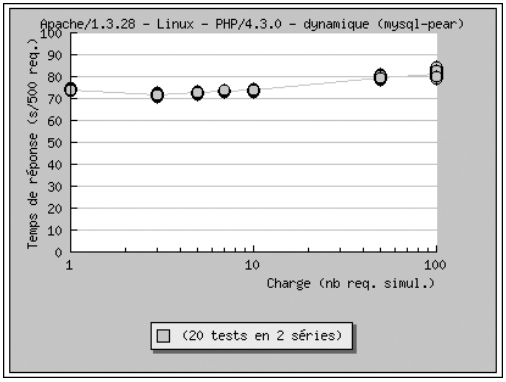


Figure 19.5 : *Temps de réponse d'une page dynamique sans optimisation*

Tableau 19.2 : Temps de réponse sans optimisation d'une page dynamique

Nombre de requêtes simultanées	Temps total d'exécution	Temps total d'exécution minimum	Temps total d'exécution maximum	Nombre de requêtes en échec
1	74.03	73.29	75.139	0
3	71.64	70.47	72.90	0
5	72.58	71.66	73.27	0
7	73.37	72.60	74.10	0
10	73.79	73.00	74.36	0
50	79.35	78.24	80.97	0
100	81.30	78.96	84.33	moyenne 12.6 minimum 1 maximum 66

Nous pouvons faire ici la même remarque que précédemment concernant les requêtes émises une à une. Là encore, sans surprise, le temps total est plus long que lorsque les requêtes sont traitées simultanément.

Les temps d'exécution sont, cette fois-ci, plus sensibles à la charge. À tel point que le serveur et/ou la base de données ne suivent plus lorsque l'on atteint les 100 requêtes simultanées. Bien qu'ils soient représentés sur le graphe, les temps mesurés ne sont alors plus vraiment représentatifs (il est plus rapide de dire "Non... je ne peux pas traiter ta requête" que de la traiter).

L'impact de la charge se fait ressentir au-delà des 10 requêtes simultanées.

Les mesures ne laissent apparaître qu'une faible variation (de l'ordre de 1 %) d'un test à l'autre.

Comparatif avec les tests des éditions précédentes
Les tests précédents donnaient également des résultats moins bons: de l'ordre de 91 secondes contre 73 ici: soit un gain de 25%.

Page mathématique

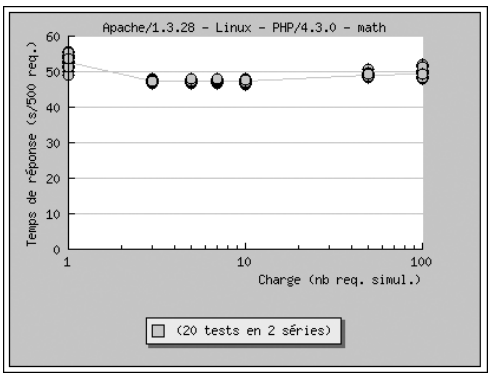


Figure 19.6 : *Temps de réponse d'une page mathématique sans optimisation*

Tableau 19.3 : Temps de réponse sans optimisation d'une page mathématique

Nombre de requêtes simultanées	Temps total d'exécution	Temps total d'exécution minimum	Temps total d'exécution maximum	Nombre de requêtes en échec
1	52.71	49.05	55.59	0
3	47.22	46.70	47.84	0
5	47.41	46.63	47.96	0
7	47.37	46.73	47.95	0
10	47.31	46.47	47.94	0
50	49.11	48.28	50.55	0
100	49.36	48.02	51.88	0

Nous ne reviendrons plus sur l'analyse des résultats obtenus lorsque les requêtes sont émises les unes après les autres pour nous concentrer sur le phénomène de charge.

Tout comme avec le script précédent, ce test est sensible à la charge. Ainsi, le temps de réponse augmente au fur et à mesure qu'augmente la charge (de 3 à 100 requêtes par secondes) exception faite d'un petit "accident" pour 5 requêtes par secondes. Cependant, cette fois, même avec 100 requêtes par secondes, aucune requête ne tombe en erreur (laissant entendre que le rôle de la base de données dans la perte de tenue en charge n'est pas négligeable). Le temps total d'exécution semble même atteindre un palier de seulement 4 % supérieur à la plus petite mesure relevée (pour trois accès concurrents).

Là encore, la stabilité des résultats obtenus est satisfaisante (des variations de plus ou moins 2 % à peine).

Comparatif avec les test des éditions précédentes
Surprise! Les tests précédents donnaient de meilleurs résultats: de l'ordre de 42 secondes contre 47 ici: soit une perte de 10%.

Reste maintenant à voir ce que peuvent nous apporter les solutions d'optimisation.

19.4. Avec Zend Optimizer

Entre les versions 3 et 4 de PHP, il y a, semble-t-il, eu un gros effort concernant l'optimisation du compilateur PHP. Cependant, cela n'a pas empêché la société Zend de proposer (gratuitement) un optimisateur de code appelé Zend Optimizer.

Description

Cet optimisateur a pour objectif d'effectuer quelques traitements supplémentaires sur le code généré, afin de le rendre encore plus rapide à l'exécution. Il va donc de soi que l'utilisation de

l'optimisateur ne sera bénéfique que si le temps gagné à l'exécution est supérieur au temps perdu à effectuer le travail d'optimisation.

Ce code est disponible gratuitement à l'adresse http://www.zend.com (en anglais).

pour les systèmes d'exploitation :

- Windows ;
- Linux glibc2.1 ;
- FreeBSD 3.4 et 4.0 ;
- Solaris Sparc ;
- IBM AIX Server.
- Mac OS X (pour PHP>=4.3.2)

La version testée est la version 2.1.0 pour Linux compatible avec les versions de PHP>=4.0.5.

Version 2.5
La dernière version en date est la 2.5, elle supporte officiellement les versions de PHP allant de la 4.0.5 à la 5.0.0RC2.

Installation

Son installation est très simple.

Sous Windows

Après avoir téléchargé la version de Zend Optmizer gratuitement sur le site officiel (http://www.zend.com), il suffit de lancer son installation et de suivre les étapes décrites ci-dessous.

Tout d'abord, vous serez invités à lire et à accepter la licence.

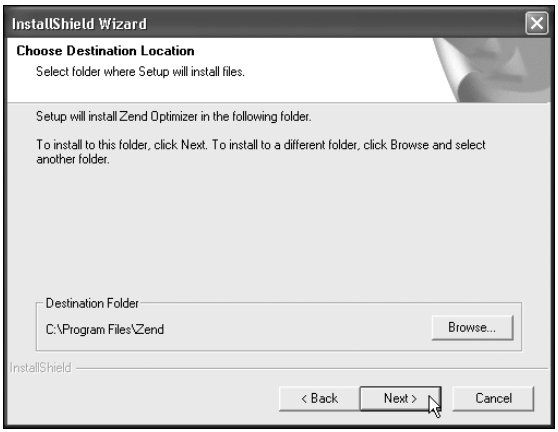


Figure 19.7 :
Sélection du répertoire d'installation

Puis vous pourrez sélectionner le répertoire dans lequel vous souhaitez installer le logiciel.

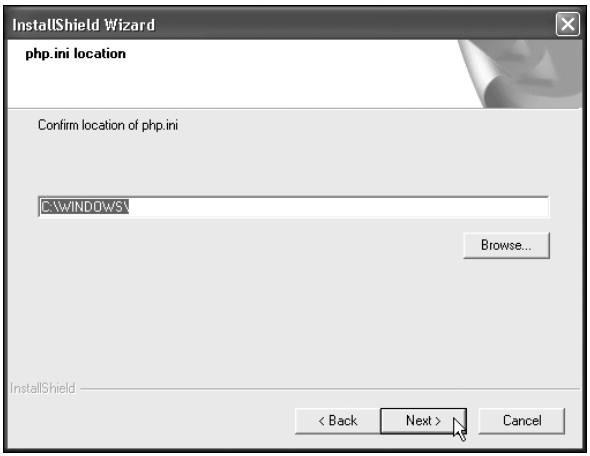


Figure 19.8 : *Sélection du répertoire php.ini*

Ensuite, vous devrez confirmer ou modifier le chemin du répertoire contenant le fichier *php.ini* utilisé par votre serveur. Une fenêtre vous précisera alors que l'ancien fichier *php.ini* a été sauvegardé sous un autre nom.

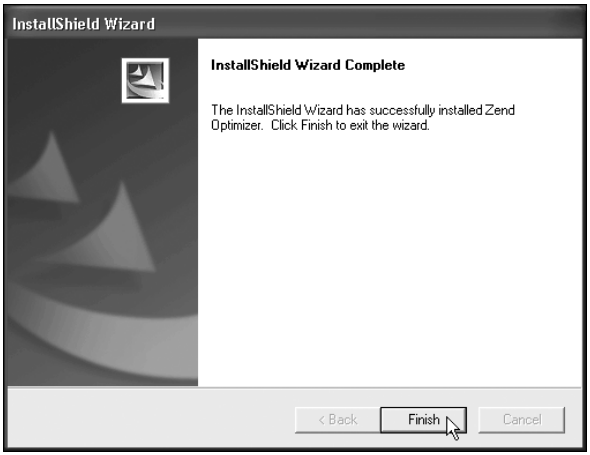


Figure 19.9 : *C'est installé*

Et voilà c'est fait.

Sous Linux

Il vous suffit de copier le fichier dans un espace temporaire (ex. : */tmp*) et de le décompresser. Vous aurez certainement besoin d'être connecté en tant qu'administrateur (`root`) afin de pouvoir créer l'arborescence de Zend Optimizer (typiquement sous */usr/local*).

```
# gunzip ZendOptimizer-2.1.0b-Linux_glibc21-i386.tar.gz
# tar xvf ZendOptimizer-2.1.0b-Linux_glibc21-i386.tar
# cd ZendOptimizer-2.1.0b-Linux_glibc21-i386
```

Il suffit maintenant de lancer le script d'installation :

```
# ./install.sh
```

et de répondre correctement aux questions...

Figure 19.10 :
Fenêtre d'accueil

...après avoir dit bonjour. Vous l'aurez compris, vous n'avez qu'à taper Entrée.

Vous êtes ensuite invités à lire et accepter (ou non) la licence.

Figure 19.11 :
Chemin d'installation

Indiquez ensuite le nom du répertoire dans lequel vous souhaitez installer Zend Optimizer. Par défaut, le script vous propose "*/usr/local/Zend*". N'hésitez pas à le changer pour "*/usr/local/Zend_Optimizer_2.1.0*". C'est fait ? Tapez sur Entrée.

Figure 19.12 :
Chemin du fichier php.ini

Vous êtes maintenant invité à indiquer le chemin du répertoire contenant votre fichier *php.ini*. Par défaut, il s'agit du répertoire "*/usr/local/lib*". C'est bon ? Passons à la suite en tapant Entrée.

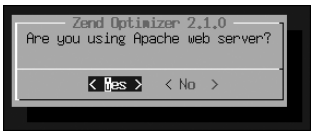


Figure 19.13 :
Sélection du serveur web

Quelque peu indiscret, Zend Optimizer vous demandera si vous utilisez un serveur Apache ou non. Nous vous laissons répondre en votre âme et conscience. Toutefois, l'histoire ne dit pas ce qu'il se passe si l'on répond "No". Si vous avez la même configuration que nous (ce qui est assez probable) vous aurez laissé la surbrillance sur "Yes" et tapé Entrée.

Figure 19.14 :
Chemin du serveur web

Une réponse appelant une question, vous devez à présent indiquer le chemin du répertoire `bin/` d'Apache (peut-être */usr/local/apache/bin*). Ne faiblissons pas... et effectuons une nouvelle pression sur la touche Entrée.

Figure 19.15 :
Déplacement de php.ini

Cette fois, c'est pour nous signaler que le fichier *php.ini* a été déplacé de son ancienne position (hum ! oui, dans notre cas, il était sous */usr/local/Zend/etc/* mais, habituellement, il se trouve sous */usr/local/lib*) vers la nouvelle position */usr/local/Zend_Optimizer_2.1.0/etc/*, et un lien symbolique a été créé de l'ancienne vers la nouvelle. Une fois que vous avez pris connaissance de cette information, vous pouvez taper Entrée.

Figure 19.16 :
Fenêtre de fin

Et voilà... c'est prêt. C'est quasiment le dernier appui sur la touche Entrée.

Figure 19.17 :
Redémarrage d'Apache

Vous êtes maintenant invité à relancer le serveur Apache puis une page vous confirme le succès (ou non) de l'opération.

Vérification

Une fois l'opération d'installation effectuée, vous pouvez vérifier qu'elle s'est bien déroulée en appelant un script contenant simplement la ligne `<?php phpinfo(); ?>`. Vous devrez alors apercevoir le texte suivant :

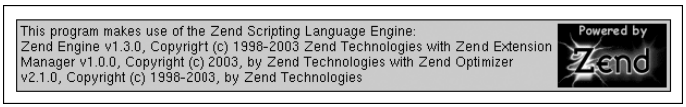


Figure 19.18 : *phpinfo()*

L'indication "*with Zend Optimizer*" confirme le succès de l'opération d'installation.

Vous constaterez également la présence de six nouvelles lignes dans votre fichier *php.ini*.

```
[Zend]
zend_optimizer.optimization_level=15
zend_extension=<chemin vers Zend Optimizeur>/lib/ZendOptimizer.so
zend_extension_manager.optimizer=<chemin>
zend_extension_manager.optimizer_ts=<chemin>
zend_extension_ts=<chemin>
```

Si vous souhaitez désactiver l'optimisateur Zend, il vous suffira de mettre ces quelques lignes en commentaire (en les faisant précéder d'un point-virgule).

Mesures

Page quasi-statique

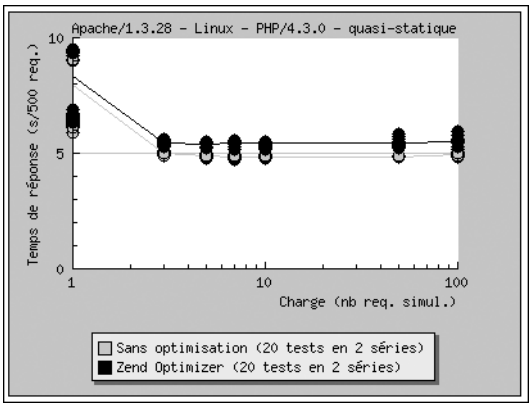


Figure 19.19 : *Temps de réponse d'une page quasi-statique avec Zend Optimizer*

Tableau 19.4 : Temps de réponse avec Zend Optimizer d'une page quasi-statique

Nombre de requêtes simultanées	Temps total d'exécution (en secondes)	Temps total d'exécution minimum (en secondes)	Temps total d'exécution maximum (en secondes)	Nombre de requêtes en échec
1	8.32	6.34	15.71	0
3	5.46	5.29	5.59	0
5	5.41	5.20	5.50	0
7	5.42	5.18	5.55	0
10	5.42	5.17	5.52	0
50	5.47	5.19	5.85	0
100	5.52	5.27	5.94	0

Comme en l'absence de solution d'optimisation, les résultats sont quasiment indépendants de la charge (tout juste 2 % de plus que le minimum avec 100 requêtes simultanées).

Les résultats obtenus ont peu varié d'un test à l'autre.

Mais, malheureusement, le résultat obtenu n'est absolument pas celui attendu. En effet, les temps de réponse ont été augmentés de 12 % par rapport à une solution sans cache. "C'est quoi cette histoire?" nous direz-vous. En fait, il suffit de se rappeler le principe de fonctionnement de Zend Optimizer pour bien comprendre : comme cela a été indiqué, Zend Optimizer retravaille le code généré pour l'optimiser et faire qu'il s'exécute plus vite. Or, dans notre script de test, il y a relativement peu de code et probablement peu de solutions d'optimisation. En conséquence, Zend Optimizer perd du temps à chercher à optimiser un code qui, semble-t-il, ne peut pas l'être. D'où des temps de réponses plus importants.

Comparatif avec les tests des éditions précédentes
Les tests précédents (basés sur PHP 4.2.1 et Zend Optimizer 1.3.1) donnaient des résultats moins bons: de l'ordre de 6,6 secondes contre 5,4 ici: Le gain obtenu en un an est donc de 22%.

Page dynamique

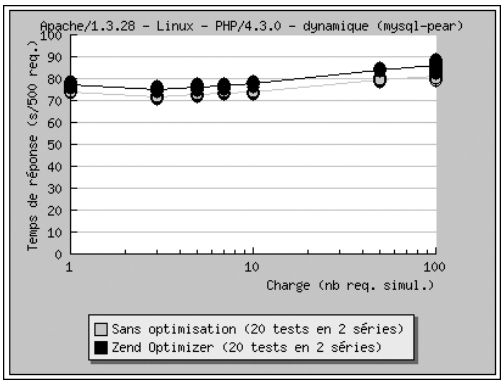


Figure 19.20 : *Temps de réponse d'une page dynamique avec Zend Optimizer*

Tableau 19.5 : Temps de réponse avec Zend Optimizer d'une page dynamique

Nombre de requêtes simultanées	Temps total d'exécution (en secondes)	Temps total d'exécution minimum (en secondes)	Temps total d'exécution maximum (en secondes)	Nombre de requêtes en échec
1	77.18	75.78	79.00	0
3	75.07	73.88	76.49	0
5	76.15	74.58	77.53	0
7	76.71	74.75	77.64	0
10	77.66	76.03	78.63	0

Nombre de requêtes simultanées	Temps total d'exécution (en secondes)	Temps total d'exécution minimum (en secondes)	Temps total d'exécution maximum (en secondes)	Nombre de requêtes en échec
50	83.73	83.23	84.83	0
100	85.84	82.48	88.62	Moyenne 14.35 Minimum 7 Maximum 69

Les temps de réponse sont, cette fois, dépendants de la charge (comme ce pouvait être le cas sans optimisateur). Ceci est plus particulièrement sensible au-delà de 10 requêtes par secondes. Finalement, nous avons des requêtes en erreur lorsque 100 requêtes sont envoyées simultanément (dans ce cas, le temps total d'exécution n'est plus significatif).

Les résultats obtenus restent légèrement supérieurs à ceux obtenus en l'absence d'optimisation. Nous pouvons donc supposer que, dans ce cas, le gain en temps d'exécution que procure le passage de l'optimisateur Zend est totalement absorbé par le temps supplémentaire nécessité pour cette opération.

Les résultats obtenus sont relativement stables d'un test à l'autre.

Comparatif avec les tests des éditions précédentes
Les tests précédents (basés sur PHP 4.2.1 et Zend Optimizer 1.3.1) donnaient des résultats moins bons: de l'ordre de 91 secondes contre 77 ici: Le gain obtenu en un an est donc de 18%.

Page mathématique

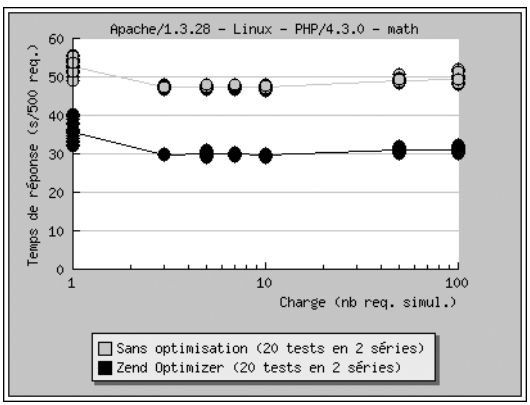


Figure 19.21 : *Temps de réponse d'une page mathématique avec Zend Optimizer*

Tableau 19.6 : Temps de réponse avec Zend Optimizer d'une page mathématique

Nombre de requêtes simultanées	Temps total d'exécution (en secondes)	Temps total d'exécution minimum (en secondes)	Temps total d'exécution maximum (en secondes)	Nombre de requêtes en échec
1	35.78	32.15	40.33	0
3	29.83	29.58	30.11	0
5	30.00	29.05	31.08	0
7	29.93	29.30	30.48	0
10	29.70	29.14	30.16	0
50	30.88	29.84	32.10	0
100	31.12	29.87	32.30	0

Là encore, le temps de réponse dépend de la charge. Mais, pour autant, le temps d'exécution subit peu l'impact de la charge (au pire + 5 %) et, surtout, aucune requête n'est rejetée, même pour 100 requêtes simultanées.

En revanche, pour une fois, le temps moyen d'exécution est nettement inférieur à ceux obtenus sans optimisation, puisque le gain est de 36 %. Ceci tend à prouver qu'effectivement Zend Optimizer joue un rôle d'optimisateur (ce dont il était possible de douter étant donné les premiers résultats). Ce script de test incluant de nombreuses boucles et opérations, le temps passé à optimiser le code a donc été utile et a largement contribué à en accélérer l'exécution.

Cependant, les valeurs sont restées relativement stables d'une série de test à l'autre.

Comparatif avec les tests des éditions précédentes
Les tests précédents (basés sur PHP 4.2.1 et Zend Optimizer 1.3.1) donnaient des résultats moins bons: de l'ordre de 36 secondes contre 30 ici: Le gain obtenu en un an est donc de 20%.

Conclusion

Si l'on considère que les résultats obtenus dans cette configuration de test sont représentatifs, alors nous pouvons conclure que l'utilisation de Zend Optimizer ne se justifie pas (bien au contraire), excepté pour des sites très particuliers sollicitant excessivement le langage PHP.

19.5. Avec Alternative PHP Cache (APC)

Description

La solution retenue par APC consiste à mettre en cache (conserver en mémoire) le résultat de la compilation (afin de ne pas avoir à renouveler les opérations d'analyse et de compilation des scripts à chaque fois que ceux-ci sont appelés).

APC est désormais disponible gratuitement sur le site (anglais) **http://pecl.php.net/apc**. La dernière version disponible est la 2.0.4 qui supporte officiellement les versions de PHP 4.2.2, 4.2.3, 4.3.0 et 4.3.2. Elle a été conçue pour compiler sous Linux, FreeBSD, OpenBSD et MacOS 10.2

Les tests ont été réalisés avec une version de développement disponible fin juillet 2003 (supportant officiellement PHP 4.2.2, 4.2.3 et 4.3.0) qui n'est plus proposé en téléchargement mais dont vous trouverez un exemplaire sur le CD-ROM fourni.

Installation

Sous Linux

Après avoir copié l'archive (celle disponible sur le CDROM est baptisée *apc-cvs_030731.tar.gz*) dans un répertoire quelconque (disons */usr/local/src*), vous devez la décompresser :

```
# gunzip apc-cvs.tar.gz
# tar xvf apc-cvs.tar
```

Puis vous devez la compiler :

```
# cd apc-cvs
# /usr/local/bin/phpize
# ./configure --with-php-config=/usr/local/bin/php-config
```

Prendre le bon chemin
Nous supposons ici, que PHP a été installé sous /usr/local/ et que par conséquent, les commandes phpize et php-config se trouvent sous le répertoire /usr/local/bin.

```
# make
# make install
```

Cette dernière instruction affiche le répertoire dans lequel vient d'être installé le fichier *apc.so*.

Modifiez votre fichier *php.ini* afin d'ajouter les lignes suivantes :

```
[APC]
zend_extension = <chemin affiché par make install>/apc.so
```

Il existe de nombreuses options mais, comme leur nom l'indique, elles sont facultatives. Nous ne les présenterons donc pas ici.

Pour désactiver le cache APC, vous pouvez mettre en commentaire ces lignes en les faisant précéder d'un point-virgule.

Vérification

Une fois l'opération d'installation effectuée, vous pouvez vérifier qu'elle s'est bien déroulée en appelant un script contenant simplement la ligne `<?php phpinfo(); ?>`. Vous devrez alors apercevoir le texte suivant :

Figure 19.22 : *phpinfo()*

L'indication "*with APC Caching*" confirme le succès de l'opération d'installation.

Mesures

Page quasi-statique

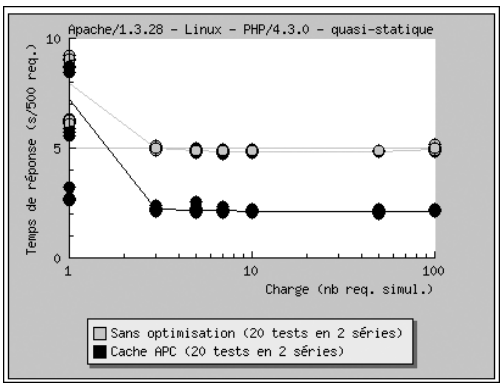


Figure 19.23 : *Temps de réponse d'une page quasi-statique avec APC*

Tableau 19.7 : Temps de réponse avec APC d'une page quasi-statique

Nombre de requêtes simultanées	Temps total d'exécution (en secondes)	Temps total d'exécution minimum (en secondes)	Temps total d'exécution maximum (en secondes)	Nombre de requêtes en échec
1	7.20	2.60	26.80	Moyenne 49,9 Minimum 0 Maximum 499
3	2.21	2.10	2.39	0
5	2.19	2.06	2.54	0
7	2.16	2.05	2.32	0
10	2.13	2.05	2.21	0
50	2.11	2.03	2.22	0
100	2.13	2.09	2.23	0

La première grosse surprise à la lecture de ces résultats (et après une rapide analyse) c'est de constater que systématiquement la première série de test est tombée en erreur. Et nous n'avons malheureusement trouvé aucune justification à ce phénomène. Nous mettrons donc cela sur le fait qu'il s'agit d'une version en cours de développement.

Si l'on fait abstraction de ce problème, nous pouvons constater que les temps de réponse sont extrêmement bons. Le gain est ici de 56 % par rapport à la version sans optimisation. APC démontre ici l'intérêt de mettre en cache le résultat de la compilation.

Comparatif avec les tests des éditions précédentes
Les tests précédents (basés sur PHP 4.2.1 et APC de juin 2002) donnaient des résultats moins bons: de l'ordre de 5.1 secondes contre 2.1 ici: Le gain obtenu en un an est donc de 59%.

Page dynamique

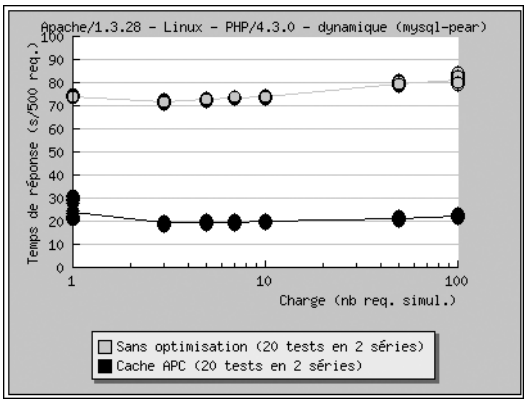


Figure 19.24 : *Temps de réponse d'une page dynamique avec APC*

Tableau 19.8 : Temps de réponse avec APC d'une page dynamique

Nombre de requêtes simultanées	Temps total d'exécution (en secondes)	Temps total d'exécution minimum (en secondes)	Temps total d'exécution maximum (en secondes)	Nombre de requêtes en échec
1	23.75	20.28	31.30	0
3	19.26	18.00	20.07	0
5	19.31	18.25	20.35	0
7	19.67	18.51	20.59	0
10	20.00	18.90	20.75	0
50	21.32	20.09	22.02	0
100	22.13	21.13	23.50	0

Le schéma est à première vue identique aux résultats obtenus précédemment pour l'exemple dynamique. Les temps de réponse se trouvent augmentés au fur et à mesure que la charge croît, par contre, chose importante, le serveur ne se trouve pas saturé et peux répondre à toutes les requêtes.

Là, encore, APC prouve son efficacité en réduisant le temps de réponse par rapport à une solution non optimisée, avec un gain de 73 % (excusez du peu).

Comparatif avec les tests des éditions précédentes
Les tests précédents (basés sur PHP 4.2.1 et APC de juin 2002) donnaient des résultats moins bons: de l'ordre de 91 secondes contre 20 ici: Le gain obtenu en un an est donc de 78%.

Page mathématique

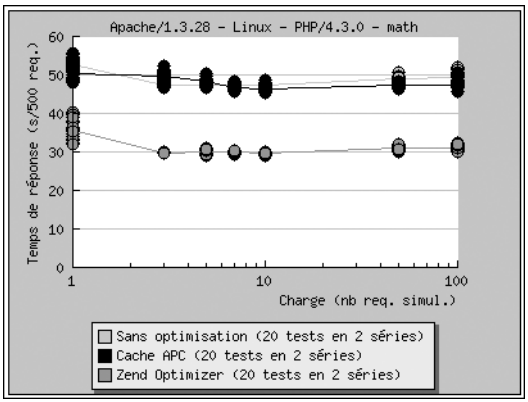


Figure 19.25 : *Temps de réponse d'une page mathématique avec APC*

Tableau 19.9 : Temps de réponse avec APC d'une page mathématique

Nombre de requêtes simultanées	Temps total d'exécution (en secondes)	Temps total d'exécution minimum (en secondes)	Temps total d'exécution maximum (en secondes)	Nombre de requêtes en échec
1	50.23	47.98	55.43	0
3	49.63	47.62	52.40	0
5	48.35	47.08	50.45	0
7	46.69	45.50	48.30	0
10	46.39	45.31	48.62	0
50	47.43	46.41	48.41	0
100	47.47	45.56	50.73	0

Apparemment, il n'est pas facile de gagner sur tous les tableaux. Alors que sur les deux tests précédents APC offre un gain en performance qui est loin d'être négligeable, au cours de ce test, les temps de réponse sont sensiblement identiques à ceux obtenus sans optimisation (et donc bien loin des résultats obtenus avec Zend Optimizer).

Il est possible d'esquisser une explication à cela. APC a beau stocker en mémoire le résultat de la compilation, il n'en reste pas moins qu'il faut exécuter ce code. Alors que Zend Optimizer réduit le temps d'exécution et offre un gain intéressant sur ce genre de script, il n'en est rien pour APC qui exécute le code obtenu par une compilation standard.


REMARQUE

Comparatif avec les tests des éditions précédentes
Les tests précédents (basés sur PHP 4.2.1 et APC de juin 2002) donnaient de meilleurs résultats: de l'ordre de 36 secondes contre 47 ici: Soit une perte de 30% en un an.

Conclusion

Même s'il est vrai qu'APC n'apporte rien en ce qui concerne le script mathématique (considéré comme non représentatif d'un site web), le gain obtenu dans les autres cas est loin d'être négligeable et justifie pleinement l'utilisation d'un système de mise en cache du code compilé.

Nous avons également pu constater que globalement de gros progrès ont été réalisés depuis la précédente édition de ce livre. Ce qui fait d'APC une solution très prometteuse.

19.6. Avec PHP Accelerator (PHPA)

Description

Tout comme APC, PHP Accelerator garde en mémoire le résultat de la compilation. C'est donc également une solution de cache, mais pas uniquement, puisque PHPA intègre aussi un optimisateur de code (comme peut le faire Zend Optimizer). Ce dernier aspect de PHPA est cependant considéré par leurs auteurs comme étant au premier stade du développement (sous-entendu : il pourrait être amélioré).

PHPA, de la société ionCube, est disponible gratuitement à l'adresse http://php-accelerator.co.uk (en anglais).

Pour la version 4.3.0 de PHP, vous le trouverez pour les systèmes d'exploitation suivants :

FreeBSD, Linux (glibc2 et glibc2.2.5), OpenBSD, Solaris.

La version testée ici est la 1.3.3r2 pour PHP4.3.0 sous linux. Elle reste à ce jour la dernière version disponible.

Installation

Sous Linux

Après avoir copié le fichier dans un répertoire donné (ex. : */tmp*), il suffit tout simplement de décompresser l'archive.

```
# gunzip php_accelerator-1.3.3r2_php-4.3.0_linux_i686-glibc2.1.3.tgz
# tar xvf php_accelerator-1.3.3r2_php-4.3.0_linux_i686-glibc2.1.3.tar
```

Il faut ensuite copier le module dans le répertoire des extensions PHP :

```
# cp php_accelerator-1.3.3r2_php-4.3.0_linux_i686-glibc2.1.3/php_accelerator-1.3
✂ .3r2.so /usr/local/lib/php/extensions/php_accelerator_1.3.3r2.so
```

puis ajouter quelques lignes au fichier *php.ini*.

```
[PHPA]
zend_extension=/usr/local/lib/php/extensions/php_accelerator_1.3.3r2.so
```

Vérification

Une fois l'opération d'installation effectuée, vous pouvez vérifier qu'elle s'est bien déroulée en appelant un script contenant simplement la ligne `<?php phpinfo(); ?>`. Vous devriez alors apercevoir le texte suivant :

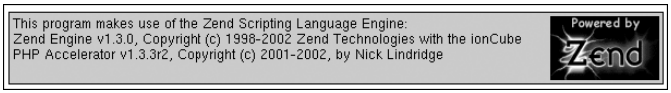


Figure 19.26 : *phpinfo()*

L'indication "*with PHP Accelerator*" confirme le succès de l'opération d'installation.

Mesures

Page quasi-statique

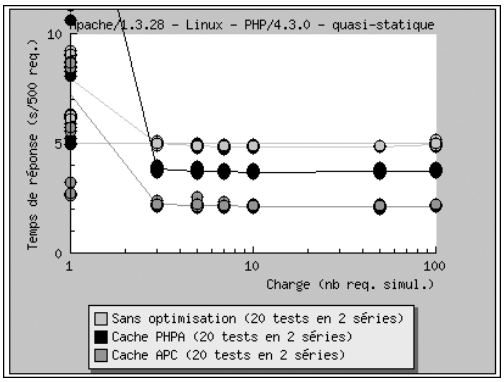


Figure 19.27 : *Temps de réponse d'une page quasi-statique avec PHPA*

Tableau 19.10 : Temps de réponse avec PHPA d'une page quasi-statique

Nombre de requêtes simultanées	Temps total d'exécution (en secondes)	Temps total d'exécution minimum (en secondes)	Temps total d'exécution maximum (en secondes)	Nombre de requêtes en échec
1	20.38	4.92	46.85	0
3	3.82	3.66	3.99	0
5	3.74	3.58	3.93	0
7	3.72	3.59	3.85	0
10	3.68	3.59	3.81	0
50	3.73	3.59	3.90	0
100	3.75	3.64	3.89	0

Les valeurs obtenues lorsque les appels se font les uns après les autres sont excessivement chaotiques, ce qui n'est pas sans rappeler le phénomène observé avec APC, si ce n'est qu'ici, les requêtes ne tombent pas en erreur.

Pour le reste, les résultats sont très bons (gain de 24%) mais pas tout à fait à la hauteur de ceux offerts par APC. La différence mesurée entre ces deux solutions étant tout de même de 40%.

Comparatif avec les tests des éditions précédentes
Les tests précédents (basés sur PHP 4.2.1 et APC 1.3.1pre3) donnaient de moins bons résultats: de l'ordre de 4.8 secondes contre 3.7 ici: Soit un gain de 23% d'une année à l'autre.

Page dynamique

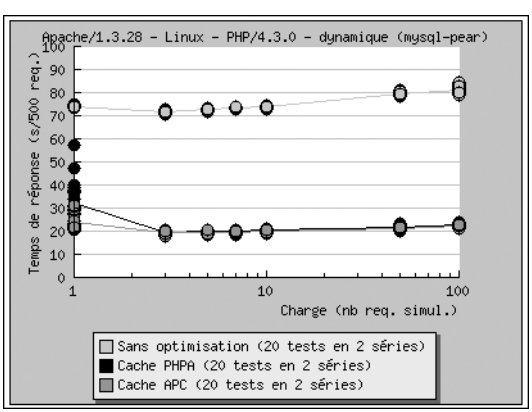


Figure 19.28 : *Temps de réponse d'une page dynamique avec PHPA*

Tableau 19.11 : Temps de réponse avec PHPA d'une page dynamique

Nombre de requêtes simultanées	Temps total d'exécution (en secondes)	Temps total d'exécution minimum (en secondes)	Temps total d'exécution maximum (en secondes)	Nombre de requêtes en échec
1	32.02	21.12	57.02	0
3	19.60	18.54	20.53	0
5	19.81	18.88	20.32	0
7	20.05	19.10	20.62	0
10	20.42	19.74	20.90	0
50	21.88	20.97	23.33	0
100	22.80	21.45	23.82	0

PHPA offre cette fois ci des résultats comparables à ceux obtenus avec APC. Et dans ce cas aussi, le nombre de requêtes en échec pour 100 requêtes simultanées est nul.

Comparatif avec les tests des éditions précédentes
Les tests précédents (basés sur PHP 4.2.1 et APC 1.3.1pre3) donnaient de moins bons résultats: de l'ordre de 42 secondes contre 20 ici: Soit un gain de 52% d'une année à l'autre.

Page mathématique

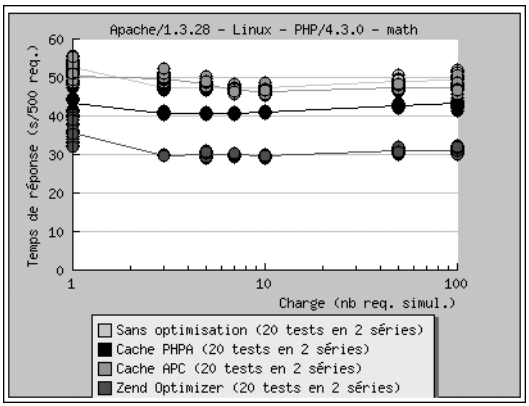


Figure 19.29 : *Temps de réponse d'une page mathématique avec PHPA*

Tableau 19.12 : Temps de réponse avec PHPA d'une page mathématique

Nombre de requêtes simultanées	Temps total d'exécution (en secondes)	Temps total d'exécution minimum (en secondes)	Temps total d'exécution maximum (en secondes)	Nombre de requêtes en échec
1	43.47	41.05	53.81	0
3	40.56	40.20	41.33	0
5	40.67	40.24	41.14	0
7	40.80	40.45	41.17	0
10	40.90	40.56	41.45	0
50	42.52	41.86	43.20	0
100	43.43	41.50	47.30	0

Sur ce test, PHPA améliore légèrement les performances obtenues avec APC sans toutefois atteindre le niveau de Zend Optimisateur.

Le gain est ici de près de 14 % par rapport à une solution non optimisée.

Comparatif avec les tests des éditions précédentes
Les tests précédents (basés sur PHP 4.2.1 et APC 1.3.1pre3) donnaient des résultats sensiblement identiques: de l'ordre de 39.8 secondes contre 40.7 ici:

Conclusion

Tout comme APC, PHPA est une solution d'optimisation gratuite et performante. Elle allie optimisation du code compilé et mise en cache du code avec une certaine efficacité, même si l'on sent que l'optimisation du code pourrait être améliorée (comme le démontre Zend Optimizer).

19.7. Avec Zend Accelerator (Zend Performance Suite)

Description

Zend Accelerator est une solution destinée aux professionnels (donc payante), qui est basée sur le même principe que PHPA. Elle est la combinaison d'un optimisateur de code (Zend Optimisateur) et d'un cache (la spécificité de Zend Accelerator).

Elle est disponible pour les systèmes d'exploitation :

- Linux glibc2.1 ;
- FreeBSD 4.3 ;
- Solaris Sparc.

Nous avons ici testé la version d'évaluation de Zend Accelerator 3.5.0 pour Linux, disponible sur le site (anglais) **http://www.zend.com**.

La dernière version disponible est la 4.0, c'est officiellement la seule des solutions présentées ici, à supporter PHP 5.0 (et toutes les versions de PHP à partir de PHP 4.1.2)

Installation

Son installation est très simple.

Sous Linux

Il vous suffit de copier l'archive dans un espace temporaire (ex. : */tmp*) et de la décompresser. Vous aurez certainement besoin d'être connecté en tant qu'administrateur (`root`) afin de pouvoir créer l'arborescence de Zend Accelerator (typiquement sous */usr/local*).

```
# gunzip ZendPerformanceSuite-3.5.0-Linux_glibc21-i386.tar.gz
# tar xvf ZendPerformanceSuite-3.5.0-Linux_glibc21-i386.tar
```

Il vous suffit ensuite de lancer le script d'installation :

```
# cd ZendPerformanceSuite-3.5.0-Linux_glibc21-i386
# ./install.sh
```

et de répondre aux questions...

Figure 19.30 : *Fenêtre d'accueil*

... après avoir dit "bonjour". Vous pouvez taper Entrée pour passer aux choses sérieuses.

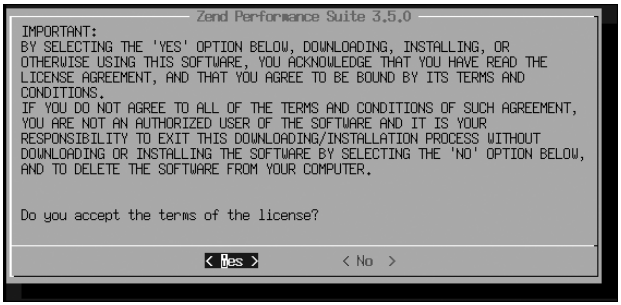


Figure 19.31 : *Licence*

Acceptez vous les termes de la licence? Nous vous laissons répondre en votre âme et conscience. Nous, nous avons opté pour l'option "Yes".

Figure 19.32 : *Clé d'activation*

Si le script d'installation ne trouve pas de clé d'activation, celui-ci vous propose d'indiquer où le fichier peut se trouver (il s'agit d'un fichier *zend_accelerator.dat*, habituellement sous le répertoire *data*). Vous pouvez également la télécharger (c'est ce que nous avons fait pour obtenir une licence d'évaluation).

Intranet

La licence doit être téléchargée depuis le poste où doit être installé Zend Accelerator. Il n'est, par exemple, pas possible de lancer le script d'installation depuis un poste relié à Internet pour utiliser la clé d'activation sur un poste en Intranet. Si votre poste n'est pas relié à Internet, sélectionnez tout de même l'option "download a license file from www.zend.com". Un message d'erreur vous indiquera alors comment vous y prendre pour récupérer manuellement un fichier de licence.

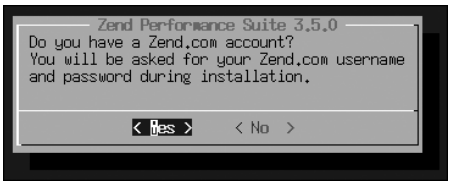


Figure 19.33 : *Compte Zend*

Il vous est ensuite demandé si vous possédez un compte Zend. A priori la réponse est "oui" puisque pour pouvoir télécharger Zend Accelerator, vous avez dû ouvrir (gratuitement) un compte sur le site. Tapez donc Entrée.

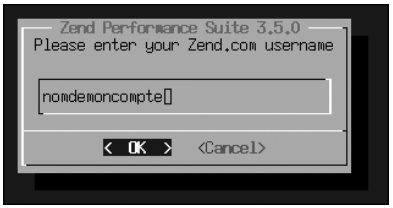


Figure 19.34 : *Saisie de l'identifiant*

Vous devez alors, ici, saisir l'identifiant de votre compte.

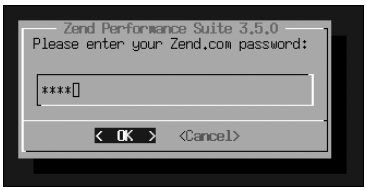


Figure 19.35 :
Saisie du mot de passe

Saisissez ensuite le mot de passe.

Figure 19.36 :
Type de la licence

Il vous est ensuite demandé quel est le type de la licence. Dans notre cas, il s'agit d'une licence d'évaluation.

Figure 19.37 :
Téléchargement

Et voilà... le fichier de licence est téléchargé. Tapez sur la touche Entrée.

Figure 19.38 :
Chemin d'installation

Vous êtes alors invité à saisir le chemin du répertoire où vous souhaitez installer Zend Accelerator. Nous vous suggérons */usr/local/Zend_Performance_Suite-3.5.0* (même si, par défaut, le script vous proposera */usr/local/Zend*).

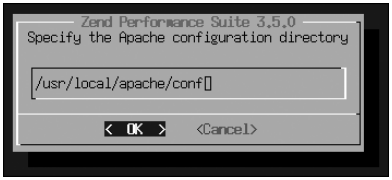


Figure 19.39 :
Chemin du répertoire du fichier de configuration du serveur

Puisqu'il veut vraiment tout savoir, il vous faut maintenant donner le chemin du répertoire hébergeant le fichier de configuration d'Apache (probablement */usr/local/apache/conf*). C'est bon ? Allez zou !... Entrée.

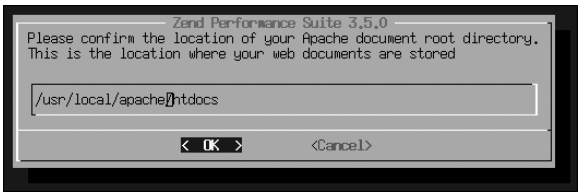


Figure 19.40 : *Répertoire racine du serveur*

Pas avare de questions, le script vous demande maintenant de préciser quel est le chemin du répertoire constituant la racine de votre serveur web (probablement */usr/local/apache/htdocs*). Ceci va lui servir pour installer les scripts d'administration de Zend Accelerator (on ne se refuse rien).

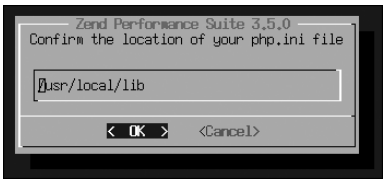


Figure 19.41 : *php.ini*

Il est temps désormais de préciser le chemin du répertoire hébergeant le fichier *php.ini* (généralement */usr/local/lib*). Une nouvelle pression sur Entrée et nous devrions être bientôt débarrassés.

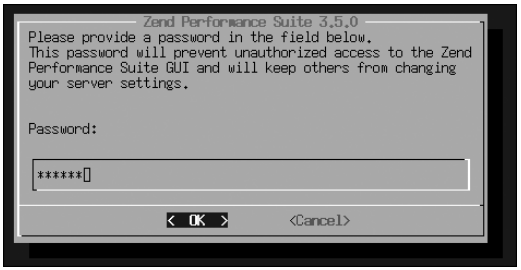


Figure 19.42 : *Mot de passe de l'outil d'administration*

Le script vous demande alors de saisir un mot de passe qui servira à protéger l'accès à la page d'administration de Zend Accelerator.

Figure 19.43 : *Confirmation du mot de passe*

Comme c'est l'usage, vous êtes invité à confirmer votre mot de passe (pour s'assurer qu'aucune erreur de saisie n'est intervenue).

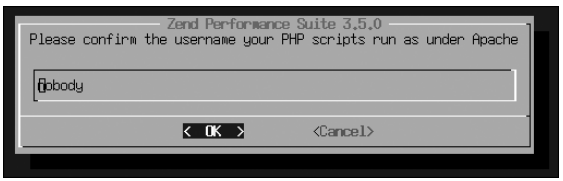


Figure 19.44 : *Confirmation de l'utilisateur Apache*

Puis, il vous faudra confirmer le nom d'utilisateur sous lequel tourne le serveur Apache.

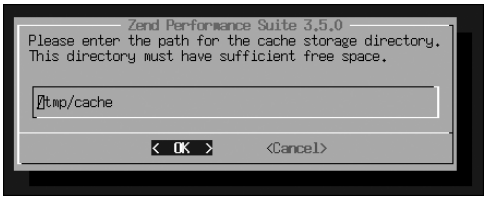


Figure 19.45 : *Espace cache*

Vous devrez également spécifier un espace de stockage temporaire des fichiers de cache.

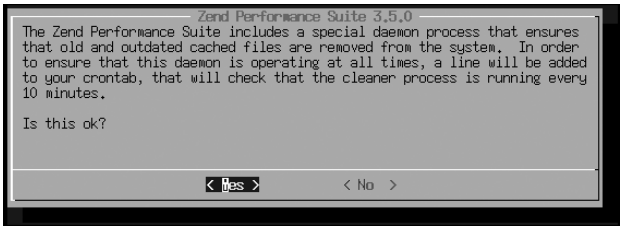


Figure 19.46 : *Crontab*

Il vous est également demandé d'accepter d'ajouter une entrée dans la crontab afin de pouvoir effectuer un nettoyage régulier du contenu obsolète du cache.

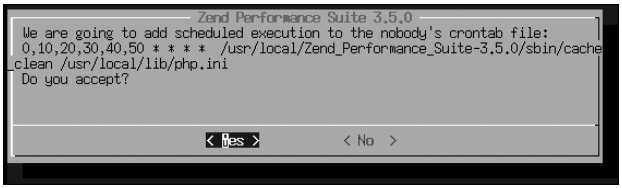


Figure 19.47 : *Confirmation crontab*

Pour ne rien vous cacher, Zend Accelerator précise la ligne qui sera ajoutée à la crontab avant de vous demander confirmation.

Figure 19.48 : *Lien symbolique*

Il semblerait que nous en ayons fini avec les questions. Le script d'installation vous indique alors que votre fichier *php.ini* a été déplacé de son ancienne position (ici */usr/local/Zend/etc/*

mais, habituellement, */usr/local/lib*) vers sa nouvelle : */usr/local/Zend_Accelerator_2.0.2/etc*. Un lien symbolique est alors créé de l'ancienne position vers la nouvelle. C'est noté ? Entrée.

Figure 19.49 : *Fenêtre de fin*

Voilà, c'est fini... ou presque.

Figure 19.50 : *Relance du serveur*

Il est temps de relancer le serveur web afin de prendre en compte les changements.

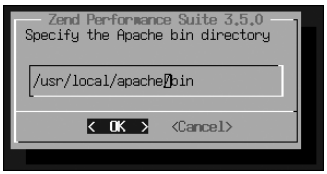


Figure 19.51 : *Fenêtre de fin*

...mais pour cela, Zend Optimizer doit connaître l'emplacement du répertoire *bin*/ d'Apache.

Figure 19.52 : *Fenêtre de fin*

Cette c'est bon.

Vérification

Une fois l'opération d'installation effectuée, vous pouvez vérifier qu'elle s'est bien déroulée en appelant un script contenant simplement la ligne `<?php phpinfo(); ?>`. Vous devriez alors apercevoir le texte suivant :

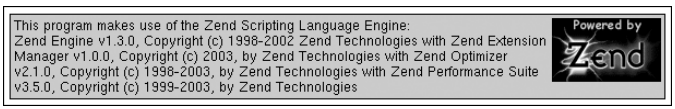


Figure 19.53 : *phpinfo()*

L'indication "*with Zend Accelerator*" (ajoutée à "*with Zend Optimizer*") confirme le succès de l'opération d'installation.

Vous constaterez également la présence de nombreuses nouvelles lignes dans votre fichier *php.ini*.

```
[Zend]
zend_gui_password=<le mot de passe crypté de l'interface d'administration>
zend_accelerator.use_blacklist_filename=<chemin vers Zend
✂ Accelerator>/etc/user_blacklist.ZendAccelerator.txt
zend_accelerator.validate_timestamps=1
zend_accelerator.use_cwd=1
zend_extension=<chemin vers Zend Accelerator>/lib/ZendExtensionManager.so
...etc...
```

Si vous souhaitez désactiver l'optimisateur Zend, il vous suffira de mettre ces lignes en commentaire (en les faisant précéder d'un point-virgule).

Mesures

Page quasi-statique

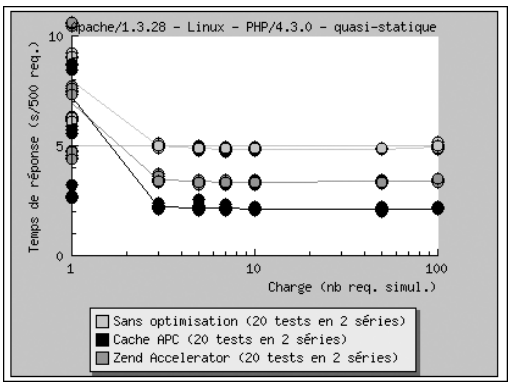


Figure 19.54 : *Temps de réponse d'une page quasi-statique avec Zend Accelerator*

Tableau 19.13 : Temps de réponse avec Zend Accelerator d'une page quasi-statique

Nombre de requêtes simultanées	Temps total d'exécution (en secondes)	Temps total d'exécution minimum (en secondes)	Temps total d'exécution maximum (en secondes)	Nombre de requêtes en échec
1	6.96	4.39	13.42	0
3	3.47	3.33	3.70	0
5	3.37	3.24	3.46	0
7	3.35	3.23	3.47	0
10	3.34	3.24	3.45	0
50	3.36	3.28	3.46	0
100	3.38	3.31	3.49	0

Les valeurs obtenues sont du niveau de ceux de PHPA (gain de 31%) mais n'atteignent pas ceux d'APC. En revanche les appels qui se font les uns après les autres offrent des temps de réponse relativement stable (en tout cas plus qu'avec PHPA ou APC).

Comparatif avec les tests des éditions précédentes
Lors du test précédent (basé sur PHP 4.2.1 et Zend Accelerator 2.0.2), étonnamment celui-ci ne fonctionnait pas.

Page dynamique

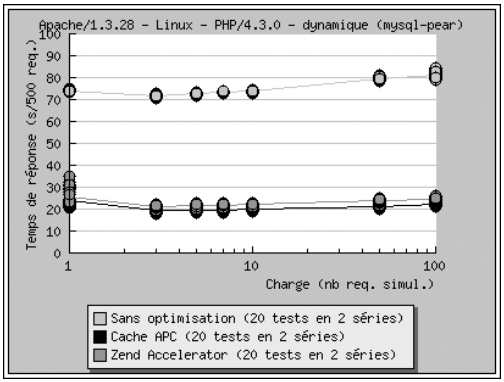


Figure 19.55 :
Temps de réponse d'une page dynamique avec Zend Accelerator

Tableau 19.14 : Temps de réponse avec Zend Accelerator d'une page dynamique

Nombre de requêtes simultanées	Temps total d'exécution (en secondes)	Temps total d'exécution minimum (en secondes)	Temps total d'exécution maximum (en secondes)	Nombre de requêtes en échec
1	25.34	22.44	35.00	0
3	21.22	19.64	22.05	0
5	21.50	20.00	22.51	0
7	21.79	20.10	22.98	0
10	22.04	22.38	22.99	0
50	23.72	21.93	25.01	0
100	24.66	23.11	26.06	0

Zend Accelerator offre une optimisation tout à fait intéressante (gain de 70%) mais toutefois légèrement moindre que celle de ses concurrents gratuits.

Encore une fois, le gain en performance a mis un terme aux échecs constaté lorsque le nombre de requêtes concurrentes atteint 100.

REMARQUE

Comparatif avec les tests des éditions précédentes
Les tests précédents (basés sur PHP 4.2.1 et Zend Accelerator 2.0.2) donnaient de moins bons résultats: de l'ordre de 28 secondes contre 21.5 ici: Soit un gain de 23% d'une année à l'autre.

Page mathématique

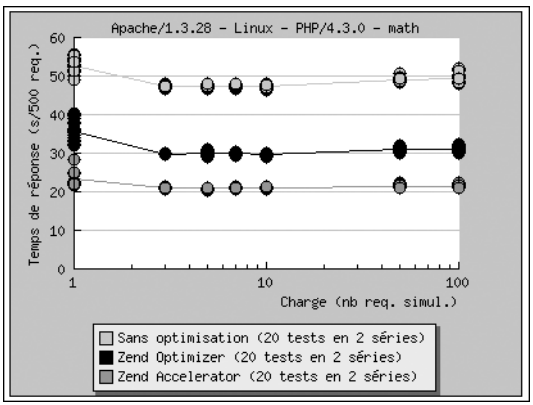


Figure 19.56 : *Temps de réponse d'une page mathématique avec Zend Accelerator*

Tableau 19.15 : Temps de réponse avec Zend Accelerator d'une page mathématique

Nombre de requêtes simultanées	Temps total d'exécution (en secondes)	Temps total d'exécution minimum (en secondes)	Temps total d'exécution maximum (en secondes)	Nombre de requêtes en échec
1	23.26	21.59	28.27	0
3	20.92	20.61	21.25	0
5	20.88	20.39	21.07	0
7	20.92	20.56	21.20	0
10	21.06	20.58	21.36	0
50	21.49	20.87	22.24	0
100	21.33	20.91	22.24	0

Les résultats obtenus ici sont également époustouflants, bien meilleurs que ceux de PHPA ou même que ceux de Zend Optimizer.

Le gain est ici de 55 % par rapport à une solution sans optimisation.

Malheureusement, rappelons-le, ce test n'est pas le plus représentative d'une page d'un site web.


REMARQUE

Comparatif avec les tests des éditions précédentes
Les tests précédents (basés sur PHP 4.2.1 et Zend Accelerator 2.0.2) donnaient de moins bons résultats: de l'ordre de 25 secondes contre 21 ici: Soit un gain de 16% d'une année à l'autre.

Conclusion

Zend Accelerator ne semble prendre l'avantage sur les autres solutions testées que dans le cas du test intégrant des boucles sur des opérations mathématiques. Ce qui semble bien maigre sachant que cette solution est payante et non les autres. Reste toutefois à compléter ces tests afin de mieux évaluer la stabilité de ces produits et l'impact d'un choix d'une autre machine et d'un autre environnement.

19.8. Conclusion sur les solutions "bas niveau" (modules PHP)

Avant de tirer une conclusion, il convient de rappeler que les tests effectués ici n'ont qu'une représentativité limitée. Les résultats peuvent en effet considérablement changer d'une configuration à l'autre. Vous n'obtiendrez certainement pas les mêmes résultats sur une machine disposant d'un processeur plus efficace, d'une plus grande quantité de mémoire ou d'un disque dur plus rapide. Mais cela dépend aussi beaucoup de l'architecture de votre système (il n'est pas vraiment conseillé de faire tourner la base de données sur la même machine que le serveur web) et, plus encore, du type de script utilisé (comme le démontre ces quelques tests). Les tests n'ont porté que sur trois scripts ; les caches n'avaient donc à gérer, au plus, que trois fichiers. Nous étions donc bien loin du cas de figure d'un véritable site hébergeant peut-être des centaines de scripts. Bref, tout cela peut faire que vous puissiez être amené à constater des différences de performance moins grandes (ou plus grandes) d'une solution d'optimisation à l'autre, voire même que la hiérarchie s'en trouve changée.

De plus, ces tests ont tous été réalisés avec la configuration par défaut. Il est possible (et même probable) que certaines des solutions testées auraient pu donner des résultats plus probants s'ils avaient été configurés pour répondre au mieux aux besoins des scripts.

Quoi qu'il en soit, la première des constatations que l'on peut faire, et ce n'est pas la moindre, c'est que ces tests confirment qu'il est possible d'améliorer grandement les performances d'un serveur web équipé de PHP. La meilleure solution consistant à optimiser le code généré et à le stocker en mémoire cache.

Enfin, il semblerait qu'il ne soit pas (sauf cas particulier) nécessaire de vider son porte monnaie pour obtenir des résultats optimum, sachant que PHP Accelerator et APC font plus que rivaliser avec Zend Accelerator. A ce jour, notre préférence se portera sur PHP Accelerator sachant que le version de production d'APC (pour les versions récentes de PHP) n'est pas encore sortie et que la version de développement en court semble contenir quelques imperfections mais cela pourrait changer à tout moment.

A contrario, Zend Optimizer ne semble pas à recommander, sauf si votre site est un peu particulier. Cette remarque n'est également pas valable si vous optez pour l'obfuscation de code (présentée dans le chapitre suivant).

Cache et mise à jour

Il y a au moins un point qui n'a pas été abordé, mais qu'il faut garder à l'esprit. Comme toujours avec les solutions de mise en cache, la difficulté consiste à déterminer la limite de validité du cache, autrement dit à répondre à la question "est-ce que le document source a été modifié depuis que je l'ai mis en cache ?". Afin de répondre à cette question, la plupart des solutions (toutes ?) contrôlent tout simplement la date de dernière modification du fichier. Vous n'aurez donc pas à arrêter puis redémarrer le serveur afin de prendre en compte les corrections apportées à vos scripts (et encore moins à aller effacer je ne sais quel fichier temporaire). Certaines des solutions d'optimisation proposent toutefois de désactiver ce contrôle afin d'augmenter encore le temps de réponse.

Même si toutes les solutions d'optimisation n'ont pas été testées ici, sachez qu'il existe très peu de logiciels de ce type pour Windows. Si malgré tout vous avez opté pour ce système d'exploitation, alors peut être devrez vous essayer Turck MM Cache disponible à l'adresse *http://www.turcksoft.com*.

19.9. Les solutions "haut niveau" (programmation PHP)

Comme nous l'avons vu dans le chapitre *Optimisation des temps de réponse* intégré au chapitre *Les en-têtes HTTP*, il est très aisé d'utiliser un système de cache de "haut niveau" consistant à stocker dans un fichier le résultat de l'interprétation PHP. En effet, ce fichier peut être rappelé à chaque nouvelle requête et/ou recréé lorsque sa durée de vie a expiré.

La principale difficulté avec ce genre de script est de déterminer la durée de vie que l'on doit donner au fichier de cache. Mais, comme il n'y a aucune règle pour fixer ce seuil, nous vous laissons vous en remettre à votre bon sens.

Une autre difficulté peut consister à déterminer les scripts qui doivent en profiter. Pour un script qui ne dépend d'aucun paramètre (`POST`, `GET`, session ou cookie), nécessite un long traitement (ou une requête SQL un peu longue), et ne varie pas toutes les 10 mn (ou qui autorise un retard de mise à jour de plus de 10 mn), il n'y a quasiment pas de questions à se poser : ce type d'optimisation s'impose. Si nous sommes dans les mêmes conditions, mais que le script dépend d'un certain nombre de paramètres, cette solution n'est envisageable que si ces paramètres ne peuvent prendre qu'un nombre fini et réduit de valeurs. Il faudra, dans ce cas, créer un fichier de cache par ensemble de paramètres possibles.

Dans tous les autres cas, nous considérons que cette solution ne doit pas être envisagée.

Si l'on considère les exemples qui nous ont servis à comparer les solutions "bas niveau", il est évident qu'un cache "haut niveau" ne peut être utilisé avec l'exemple dynamique, mais qu'en revanche, cela apporterait beaucoup à l'exemple mathématique (le délai d'expiration serait même infini... Mais bon, dans ce cas, autant utiliser une page statique).

En ce qui concerne l'exemple quasi-statique, il est plus difficile de trancher. Il y a peu de code à exécuter, ce qui fait que les temps de réponse semblent bien courts. En pareil cas, l'utilisation d'un cache de haut niveau pourrait être contre-productif. En effet, lorsque l'on fait appel à un script qui se suffit à lui-même, cela ne nécessite qu'un seul accès au disque, alors que, pour un script identique qui utilise un fichier de cache, il faut deux à trois accès au disque (un pour charger le script, un autre pour charger le fichier de cache et il faut, de plus, vérifier la date de dernière modification du script). Multiplier ainsi les accès au disque par deux ou trois peut être vraiment pénalisant, notamment vis-à-vis de la montée en charge.

En fait, il s'avère que le script qui nous a servi pour l'exemple fait une multitude d'`include`, et chaque `include` nécessite un accès au disque. Ces accès sont donc bien plus nombreux que dans une solution faisant appel à un fichier de cache.

Les performances s'en ressentent, comme le montre le graphe suivant qui compare :

- L'exemple précédent quasi-statique ;
- Sa version légèrement modifiée pour gérer un fichier de cache d'une durée de vie de 10 mn ;
- Une version statique de la page (sauvegarde au format HTML du résultat retourné par le script PHP).

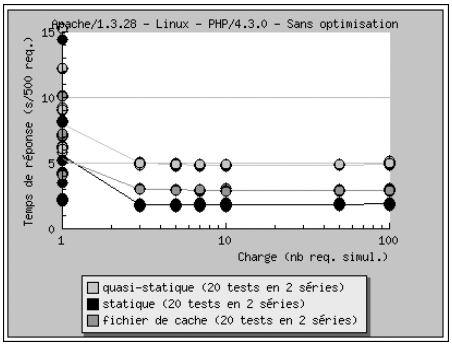


Figure 19.57 : *Comparaison des versions statique, fichier de cache et quasi-statique*

Alors que la version quasi-statique répond aux 500 requêtes en 4,8 secondes, la version avec un fichier cache répond en 2,9 secondes et la version statique, elle, ne demande que 1,8 secondes.

Comparatif avec les tests des éditions précédentes
Les tests précédents (basés sur PHP 4.2.1) donnaient de moins bons résultats: de l'ordre de 5,7 secondes contre 4,8 et 3,4 contre 2, 9 ici: Soit un gain de 15% d'une année à l'autre.

19.10. Conclusion

D'un tout autre fonctionnement que les caches de bas niveau vus précédemment, les caches de haut niveau peuvent eux aussi améliorer grandement les performances de votre site Internet, à condition toutefois que la nature des scripts demandés s'y prête.

Chapitre 20

L'obfuscation : Distribuer ses scripts sans dévoiler son code

20.1. Introduction

Si vous optez pour une licence GPL ou toute autre licence Open Source (ce que vous êtes invité à faire), la distribution de votre code ne vous causera aucun souci. Si, en revanche, vous ne souhaitez pas laisser partir votre savoir-faire dans la nature et que vous avez à distribuer une application sous forme de scripts PHP, vous devrez trouver une solution pour que le code source ne soit pas lisible, et cela sans nuire à son fonctionnement.

La solution généralement retenue s'appelle *l'obfuscation* (que l'on traduit parfois aussi "assombrissement") de code. Cette opération consiste en plusieurs points, dont les principaux sont :

1. Suppression de tous les commentaires (c'est le moins que l'on puisse faire) ;
2. Masquage de tous les noms de variables et fonctions (un appel à une fonction appelée "`a`" ou "`zedfrDS`" sera bien plus difficile à interpréter qu'un appel à une fonction appelée `connexionBaseDeDonnees()`) ;
3. Livraison du code sous sa forme compilée (c'est certainement la forme la plus difficilement interprétable pour nous autres, êtres humains).

Il existe à ce jour très peu de solutions pour PHP. Il y a, par exemple, les solutions payantes comme Zend Encoder, la toute récente ionCube PHP Encoder ou PHP guardian et une solution gratuite, POBS (PHP Obfuscator), chacune ayant ses propres caractéristiques.

D'autres solutions basées sur une autre technique existent. Ainsi Microcode et PHTML Encoder proposent des solutions consistant à crypter/décrypter les scripts PHP.

20.2. Avec Zend Encoder

Il est difficile de savoir ce que fait exactement Zend Encoder. Il n'existe en effet pas (à notre connaissance) de logiciel permettant de voir le source qui serait obtenu si l'opération était inversée. Mais l'on peut affirmer sans trop de doutes que les commentaires sont supprimés et le code livré sous sa forme compilée. Il n'est pas certain que les noms des fonctions et variables globales soient masqués, puisque les scripts ainsi obfusqués restent compatibles avec les scripts non obfusqués.

Zend Encoder est désormais intégré à Zend Safe Guard qui propose également un système de gestion de licence baptisé Zend License Manager.

Une version d'évaluation peut être téléchargée sur le site (anglais) **http://www.zend.com**.

Zend Encoder est disponible pour les systèmes d'exploitation :

- Windows (NT, 2000, XP) ;
- Linux glibc2.1 et 2.2 ;
- Solaris (2.6 et +) ;
- FreeBSD (3.4 et +).

Les scripts ainsi obfusqués ne peuvent être lus que sur les sites web disposant de Zend Optimisateur (voir chapitre *Optimisation*).


REMARQUE

Version d'évaluation
Avec la version d'évaluation les scripts générés ne sont valables que 3 jours.

La dernière version disponible à ce jour est la 3.5 qui n'est officiellement compatible qu'avec les versions 4.0.6 à 4.3 de PHP.

Installation

L'installation est très simple. Vous trouverez ci après la procédure pour la version 2.0.1

Sous Windows

Après avoir téléchargé Zend Encoder depuis le site http://www.zend.com, il suffit de lancer l'installation et de répondre à quelques questions.

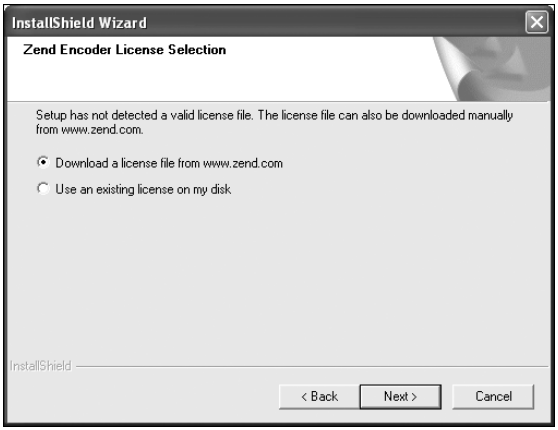


Figure 20.1 : *Installation de Zend Encoder sous Windows (1/2)*

Si le script d'installation ne trouve pas de clé d'activation, celui-ci vous propose d'indiquer où le fichier peut se trouver. Vous pouvez également la télécharger (c'est ce que nous avons fait pour obtenir une licence d'évaluation).


ATTENTION

Intranet
La licence doit être téléchargée depuis le poste où doit être installé Zend Accelerator. Il n'est, par exemple, pas possible de lancer le script d'installation depuis un poste relié à Internet pour utiliser la clé d'activation sur un poste en Intranet. Si votre poste n'est pas relié à Internet, sélectionnez tout de même l'option "download a license file from www.zend.com". Un message d'erreur vous indiquera alors comment vous y prendre pour récupérer manuellement un fichier de licence.

Figure 20.2 : *Installation de Zend Encoder sous Windows (2/2)*

Pour pouvoir télécharger Zend Encoder, vous avez dû ouvrir (gratuitement) un compte sur le site Zend. Afin de télécharger la clé d'activation vous devez, ici, saisir votre nom d'utilisateur et votre mot de passe.

Et voilà ! C'est fini.

L'interface obtenue est alors la suivante :

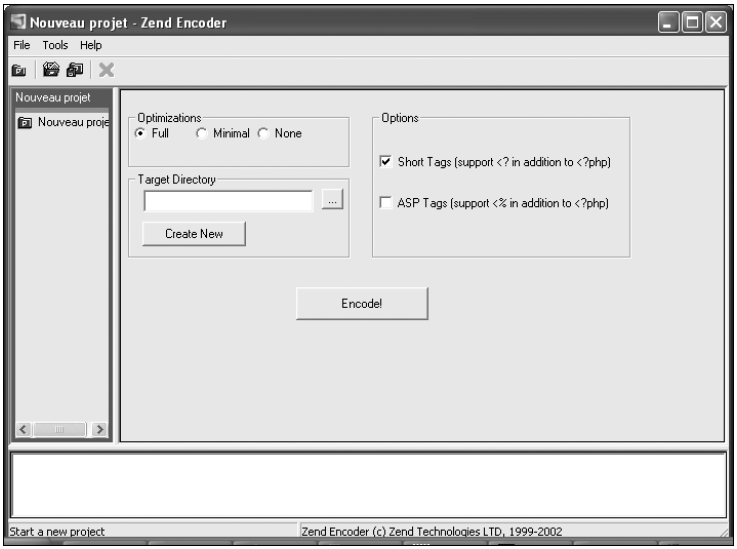


Figure 20.3 : *Interface de Zend Encoder*

Sous Linux

Vous devez copier l'archive dans un espace quelconque (ex. : */tmp*) et la décompresser.

```
# gunzip ZendEncoder-Evaluation-2.0.1-Linux_glibc21-i386.tar.gz
# tar xvf ZendEncoder-Evaluation-2.0.1-Linux_glibc21-i386.tar
```

Il suffit ensuite de lancer le script d'installation :

```
# cd ZendEncoder-Evaluation-2.0.1-Linux_glibc21-i386
# ./install.sh
```

Et c'est parti...

Figure 20.4 : *Fenêtre d'accueil*

...après avoir dit "bonjour". C'est fait ? Vous pouvez taper Entrée.

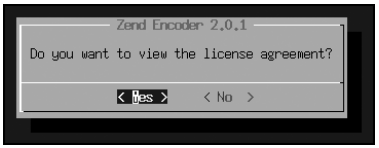


Figure 20.5 : *Licence*

Vous avez maintenant la liberté de lire ou non la licence. Nous vous laissons faire votre choix. Quoi qu'il en soit, en sélectionnant "No" ou en choisissant de lire la licence puis en validant, vous vous retrouverez face à la fenêtre suivante :

Figure 20.6 : *Acceptation de la licence*

Nous vous laissons choisir en votre âme et conscience. En ce qui nous concerne, nous avons opté pour la réponse "Yes". Vous aussi ? Très bien. Tapez Entrée et passons à la suite.

Figure 20.7 : *Chemin du répertoire d'installation*

Vous êtes maintenant invité à saisir le chemin du répertoire devant accueillir Zend Encoder. Le chemin proposé est */usr/local/Zend*. Comme vous pouvez le constater, nous avons préféré choisir */usr/local/Zend_Encoder_2.0.1*, ce qui, finalement, ne s'est pas avéré être une très bonne idée (Zend Encoder s'entêtant à aller chercher l'encodeur sous */usr/local/Zend*). Conservez donc la proposition par défaut et pressez la touche Entrée.

Figure 20.8 :
Clé d'activation

Si le script d'installation ne trouve pas de clé d'activation, celui-ci vous propose d'indiquer où le fichier peut se trouver (il s'agit d'un fichier *zend_accelerator.dat*, habituellement sous le répertoire *data* ou sous */usr/local/Zend*). Vous pouvez également la télécharger (c'est ce que nous avons fait pour obtenir une licence d'évaluation).

Intranet
La licence doit être téléchargée depuis le poste où doit être installé Zend Accelerator. Il n'est, par exemple, pas possible de lancer le script d'installation depuis un poste relié à Internet pour utiliser la clé d'activation sur un poste en Intranet. Si votre poste n'est pas relié à Internet, sélectionnez tout de même l'option "download a license file from www.zend.com". Un message d'erreur vous indiquera alors comment vous y prendre pour récupérer manuellement un fichier de licence.

Figure 20.9 :
Identifiant de compte

Pour pouvoir télécharger Zend Encoder, vous avez dû ouvrir (gratuitement) un compte sur le site Zend. Afin de télécharger la clé d'activation, vous devez ici saisir votre nom d'utilisateur…

Figure 20.10 :
Mot de passe

...ainsi que votre mot de passe.

Et voilà... Une série de fenêtres s'ouvrent et se ferment pour vous indiquer que tel ou tel fichier a été installé. Et c'est fini.

Pour obfusquer un fichier ou toute une arborescence, vous n'avez qu'à lancer l'interface graphique.

```
# /usr/local/Zend/ZendEncoderGUI
```

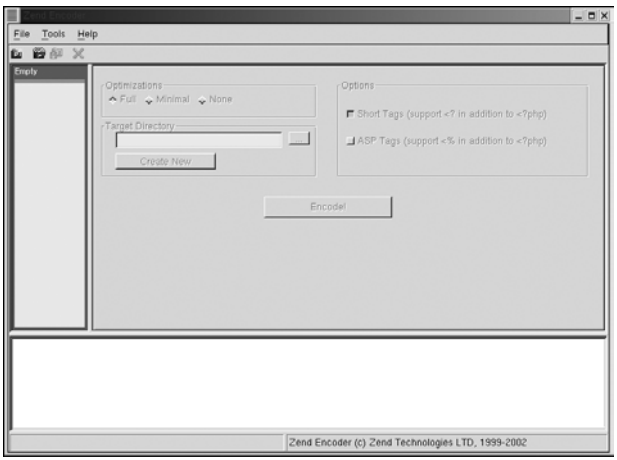


Figure 20.11 : *Interface graphique de Zend Encoder*

Utilisation

Nous avons testé la version 2.0 pour Linux.

Vous devez, dans un premier temps, créer un projet depuis le menu **File**.

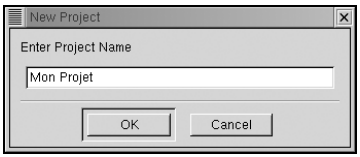


Figure 20.12 : *Nouveau projet*

Une fois le projet créé,

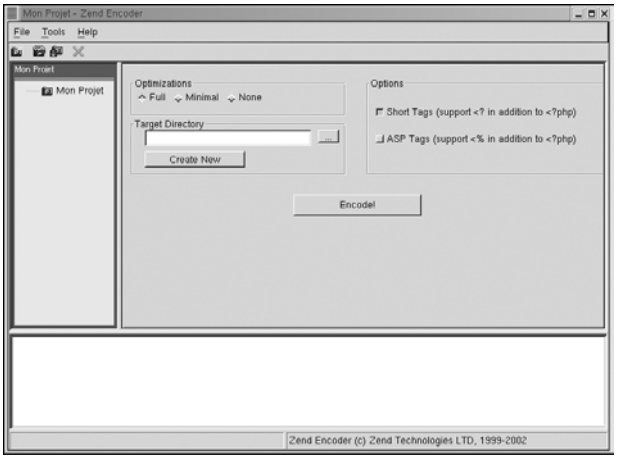


Figure 20.13 : *Interface principale*

vous devez ajouter les fichiers sources, qui constituent votre projet PHP, depuis le menu **File**.

Figure 20.14 :
Ajout de fichiers au projet

Une fois intégrés au projet, les fichiers apparaissent dans la fenêtre gauche.

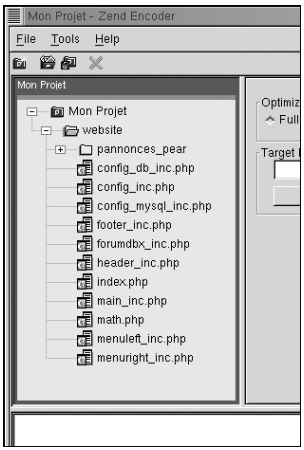


Figure 20.15 :
Répertoire source

Vous devez également indiquer le répertoire dans lequel doivent être copiés les scripts obfusqués. Pour cela cliquez sur le bouton "**...**" de **Target Directory**.

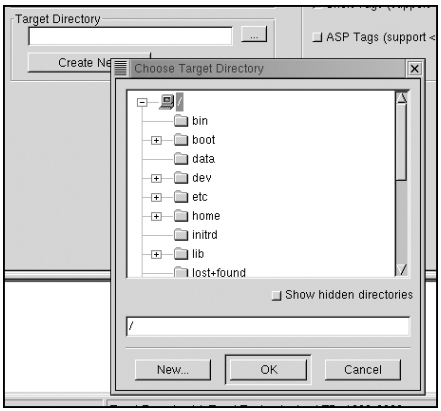


Figure 20.16 :
Répertoire destination

Une fois le projet totalement défini, il ne vous reste qu'a cliquer sur "**Encode !**".

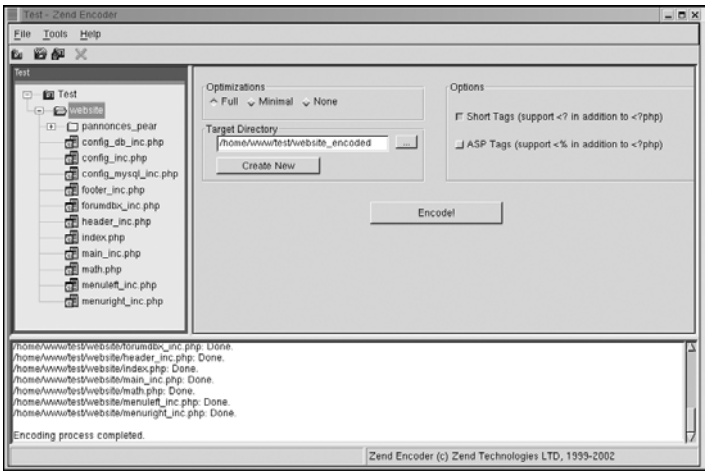


Figure 20.17 : *Obfuscation*

Et voilà, c'est fait. Nous ne vous montrerons pas le contenu d'un fichier obfusqué. C'est tellement illisible que l'on croirait voir un fichier de jurons.

Conclusion

Les fichiers assombris avec Zend Encoder sont véritablement illisibles, et ceci restera vrai tant qu'aucune solution (probablement pirate) ne permettra de passer du code compilé au code source (a priori en grande partie masqué).

Cette solution nécessite l'installation d'un logiciel sur le serveur. Mais, comme il s'agit de Zend Optimizer, cela ne constitue pas une grosse contrainte. C'est tout particulièrement vrai si vous assurez vous-même la configuration du serveur web, mais cela reste vrai également si vous passez par un hébergeur, puisqu'il n'est pas rare de voir Zend Optimizer installé.

L'utilisation de Zend Encoder est très souple, puisqu'il est possible de ne masquer qu'une partie des scripts d'un site (ce qui permet notamment de laisser lisibles les fichiers de configuration). En contrepartie, il est possible de douter du fait que les noms des fonctions ou variables globales soient masqués. Ce qui pourrait faciliter la lecture d'un script "désobfusqué", mais tout cela reste à démontrer.

20.3. Avec ionCube PHP Encoder

Tout comme dans le domaine de l'optimisation, ionCube propose une alternative au produit de Zend baptisé ionCube PHP Encoder. Le principe reste le même: il s'agit de ne fournir qu'un code précompilé illisible. Cette solution intègre également la possibilité de limiter l'exécution des scripts dans le temps et/ou par machine. Contrairement à Zend Encoder, il n'y a pas d'interface graphique permettant de sélectionner les scripts à obfusquer.

Pour fonctionner, les scripts encodés nécessitent l'installation préalable d'ionCube PHP Loader sur le serveur. Ce dernier est disponible gratuitement.

ionCube PHP Encoder n'est pas gratuite mais une version d'évaluation peut être téléchargée sur le site à l'adresse http://www.ioncube.com/. Il fonctionne sur les plateformes Linux, FreeBSD et prochainement Windows (système d'exploitation pour lequel il n'existe aujourd'hui qu'une version beta).

Nous avons bien évidemment testé la version Linux; la version d'évaluation 3.0 pour être précis.

Version d'évaluation
La version d'évaluation n'est utilisable que durant 7 jours.

Installation

Il suffit de décompresser l'archive dans son répertoire de destination (ex: */usr/local*)

```
# tar zxvf ioncube_encoder_evaluation_3.0.tar.gz
```

Utilisation

Pour crypter un répertoire complet (ici *src/*) et récupérer le résultat dans un répertoire donné (ici *dst/*) vous devrez alors taper une commande similaire à

```
# ./ioncube_encoder --key <votre cle de licence> src -o dst
```

Vous constaterez alors qu'effectivement le répertoire *dst/* contient des fichiers illisibles hormis leurs premières lignes.

Les premières lignes de ces scripts sont en fait destinées à charger en mémoire le module capable d'interpréter leur contenu, à savoir ionCube PHP Loader.

Ainsi, une des solutions possibles pour utiliser ces scripts obfusqués consiste à décompresser l'archive directement dans le répertoire contenant ces fichiers.

```
# tar zxvf ioncube_loaders_2.1.tar.gz
```

Une fois cette opération réalisée vous pouvez constater qu'il est possible d'accéder à ces scripts aussi simplement que leur équivalant non crypté.

Il existe bien évidemment, une longue liste d'options à la commande `ioncube_encoder`, mais pour répondre à votre besoin spécifique, il vaut mieux que vous vous plongiez dans les notices d'utilisation.

20.4. Avec PHP guardian

Disponible uniquement sous Windows, ce logiciel n'a pas été testé.

Il semble fonctionner sur le même principe que Zend Encoder et ionCube PHP Encoder (et offre également des options permettant de limiter la durée de vie du script ou de le limiter à une adresse IP donnée.). Il ne nécessite toutefois aucune installation sur le serveur le décodage s'effectuant par un script PHP fourni avec le logiciel.

Vous le trouverez (y compris en version d'évaluation) à l'adresse http://www.phpguardian.com/.

20.5. Avec POBS

POBS est écrit en PHP. Il fonctionne donc sur toute plateforme disposant d'un serveur web intégrant PHP.

La version testée est la version 0.99 qui reste à ce jour la dernière version disponible. Celle-ci est officiellement testée avec PHP 4.0.4 (ce qui ne signifie pas pour autant qu'elle ne fonctionne pas avec les autres versions de PHP 4)

Installation

Il suffit de décompresser, dans un coin de votre serveur web, l'archive disponible sur le site **http://pobs.mywalhalla.net/** (ou celle disponible sur le CD-ROM).

Vous serez certainement amené à modifier quelques paramètres du script *pobs-ini.inc.php*, en particulier `$SourceDir` et `$TargetDir`, qui indiquent respectivement le chemin du répertoire contenant les sources et celui destiné à recevoir les versions obfusquées des scripts.

Quoi qu'il en soit vous êtes invité à consulter le fichier *documentation-fr.txt* (profitez-en ! Pour une fois qu'il y a une notice en français…).

Utilisation

Il suffit d'appeler le script principal *pobs.php*.

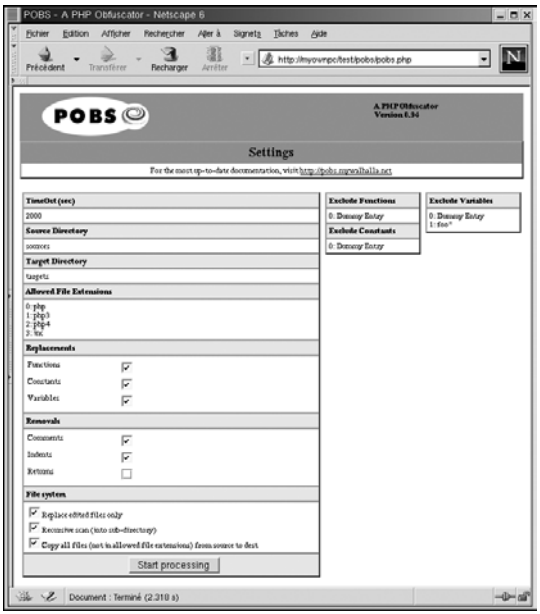


Figure 20.18 : *POBS*

L'interface qui s'offre à vous vous permet alors de modifier certains paramétrages. Vous pouvez choisir de masquer les noms des fonctions, constantes et/ou variables, et de supprimer les commentaires, indentations et/ou retours à la ligne.

Lorsque vous lancez l'obfuscation via le bouton "**Start processing**", les fichiers résultats sont alors copiés dans le répertoire désigné par le fichier *pobs-ini.inc.php* et un bilan est affiché.

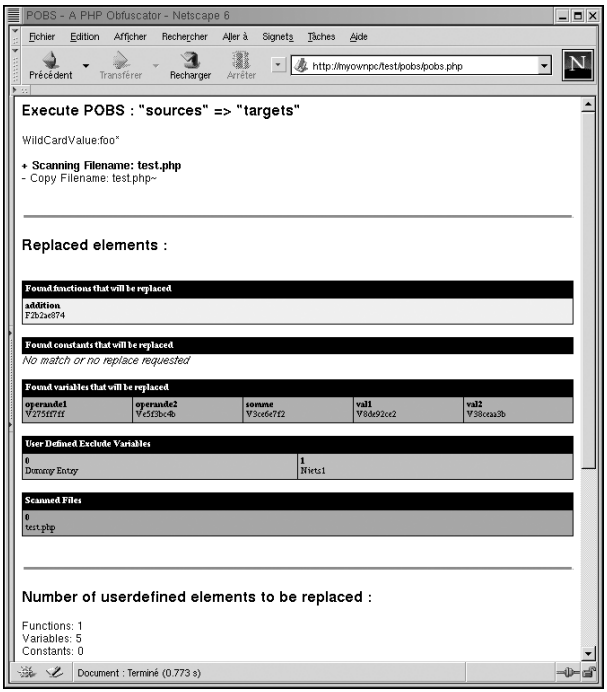


Figure 20.19 : *Bilan POBS*

Dans ce bilan apparaît la liste des fichiers traités, les noms des fonctions, constantes et variables, ainsi que leurs équivalents obfusqués.

Ainsi, l'exemple suivant :

```
<?php
  // Fonction retournant le résultat
  // de l'addition de operande1 avec operande2
  function addition($operande1, $operande2)
  {
     $somme = $operande1 + $operande2;
     return $somme;
  }
?>
<html>
<body>
<?php
   $val1 = 3;
   $val2 = 6;
   echo addition($val1, $val2);
?>
</body>
</html>
```

donne le résultat suivant (qui varie selon les options choisies) :

```
<?php
function F2b2ae874($V275ff7ff, $Ve5f3bc4b)
{$V3ce6e7f2 = $V275ff7ff + $Ve5f3bc4b;return $V3ce6e7f2;}?>
<html>
<body>
<?php
$V8de92ce2 = 3;$V38ceaa3b = 6;echo F2b2ae874($V8de92ce2, $V38ceaa3b);?>
</body>
</html>
```

Ce qui, vous en conviendrez, n'est pas très lisible (même si ici, le code étant bien simple, il n'est pas très difficile de comprendre ce que fait ce script).

Comme les noms de fonctions, constantes et variables sont modifiés par POBS, les scripts obfusqués ne pourront pas être inclus tels quels dans des scripts non obfusqués (et inversement). Si vous souhaitez conserver des scripts en clair (i.e. des fichiers de configuration), vous devrez configurer POBS (via le fichier *pobs-ini.inc.php*) afin de ne pas masquer tel ou tel nom de fonction, constante ou variable.

Conclusion

POBS a l'avantage d'être une solution gratuite offrant un bon degré de camouflage et qui ne nécessite aucun logiciel sur le serveur web. Il est donc compatible avec toutes les plateformes.

Bien que le résultat de l'obfuscation ne soit pas sous une forme compilée, le travail de restauration des sources originales est toutefois grandement facilité (parions, cependant, que les solutions permettant de transformer le résultat d'une obfuscation telle que peut en retourner Zend Encoder en un code similaire à celui que fournit POBS ne tarderont pas à apparaître. Ce type de solution perdrait alors son avantage).

Le mélange de codes sources obfusqués avec des codes sources non obfusqués n'est pas chose aisée, mais c'est peut-être là le prix à payer pour un meilleur camouflage du code.

20.6. Autres

Il existe un certain nombre d'autres solutions basées, pour la plupart, sur un simple cryptage des sources à l'aide d'un mot de passe. Toutes ces solutions nécessitent l'installation d'un logiciel spécifique sur le serveur, et ne garantissent pas nécessairement la protection du code si le mot de passe venait à être "cracké" (ou tout simplement si l'analyseur PHP était modifié pour restituer le code source décrypté qui lui est communiqué par la partie serveur du logiciel d'obfuscation).

Parmi elles, nous pouvons citer :

- PHTML Encoder disponible à l'adresse **http://www.rssoftlab.com/** ;
- Source Defendeur présenté à l'adresse **http://www.sourcedefender.com/** ;
- PHP Screw à l'adresse **http://sourceforge.net/projects/php-screw/**.

Chapitre 21

Annexe A : des exemples d'applications

21.1. Administration de bases de données

phpMyAdmin

Installation et configuration

Certains programmes d'installation automatique de PHP/MySQL comprennent déjà phpMyAdmin. En revanche, si vous avez choisi d'installer vous-même MySQL, il est fort probable que phpMyAdmin ne soit pas sur votre système, ce qui est bien dommage... En effet, phpMyAdmin est un programme libre et gratuit qui facilite grandement l'administration et l'utilisation d'un serveur de bases de données MySQL. Via une interface web simple, puissante et francisée, il permet de mener à bien quasiment toute opération relative au serveur.

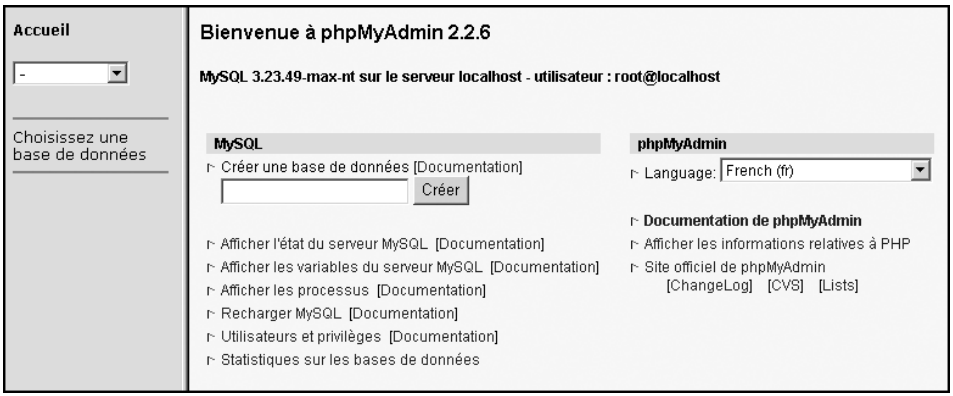


Figure 21.1 : *phpMyAdmin*

Vous pourrez ainsi créer et supprimer des bases, créer, copier et effacer des tables, lancer des requêtes SQL, charger des fichiers textes dans des tables, administrer plusieurs serveurs à la fois, sauvegarder vos données, etc.

Téléchargez la dernière version de phpMyAdmin à l'adresse **www.phpmyadmin.net** ou bien utilisez celle disponible sur le CD-ROM. Décompressez le contenu de l'archive, par exemple à la base de votre serveur web (ou dans un répertoire de votre machine pour pouvoir modifier certains scripts avant de les transférer dans un espace alloué par un hébergeur) ; cela aura pour effet de créer un répertoire *phpMyAdmin-2.5.7* (que vous pouvez éventuellement renommer en *phpmyadmin*).

À l'aide de votre éditeur de texte préféré, éditez le fichier *config.inc.php* et recherchez la ligne :

```
$cfgPmaAbsoluteUri = '';
```

Complétez avec l'adresse web à laquelle sera disponible votre phpMyAdmin. Si vous avez suivi l'exemple cité plus haut (*phpmyadmin*), vous devrez donner cette adresse :

```
$cfgPmaAbsoluteUri = 'http://www.votrenomdedomaine.com/phpmyadmin/';
```

Passons maintenant au bloc `* Server(s) configuration`. Il s'agit de configurer la connexion au serveur de bases de données. Il faut, en effet, que phpMyAdmin puisse avoir accès au serveur si l'on veut qu'il le manipule. Recherchez et complétez les lignes suivantes :

```
$cfgServers[1]['host'] = ''; // MySQL hostname
```

Complétez avec `localhost`. Si vous installez phpMyAdmin chez votre hébergeur, il peut être utile de se référer à sa documentation, le nom de l'hôte pouvant varier.

```
$cfgServers[1]['auth_type'] = 'config'; // Authentication method (config, http or
✂ cookie based)?
```

Complétez avec "`config`", "`http`" ou "`cookie`".

- `config` permet à n'importe quel utilisateur d'avoir accès à phpMyAdmin sans avoir à s'identifier. Ce sont, en effet, les paramètres du fichier *config.inc.php* qui sont utilisés. Ce choix n'est pas le plus sûr. N'importe quel utilisateur qui connaît l'adresse de votre phpMyAdmin pourrait se connecter au serveur MySQL et supprimer bases ou tables sans problème. Les options "`http`" ou "`cookie`" requièrent une authentification de la part de l'utilisateur qui se connecte.
- `http` : un formulaire demande à l'utilisateur ses login et mot de passe, puis les enregistre dans un trousseau de clés (selon les navigateurs). Ne fonctionne qu'avec PHP installé comme module Apache. PHP en CGI ne permet pas d'utiliser ce mode de connexion.
- `cookie` : les informations de connexion seront conservées dans un cookie qui ne sera valable que le temps de la session.

```
$cfgServers[1]['user'] = 'votre_pseudo_administrateur';
```

Complétez avec le pseudonyme de l'administrateur du serveur de bases de données. Généralement, c'est "`root`", à moins que vous ne l'ayez changé manuellement.

```
$cfgServers[1]['password'] = ''; // MySQL password (only needed with 'config'
✂ auth)
```

Comme vous l'indique le fichier de configuration, vous n'avez à renseigner cette ligne que si vous utilisez la méthode de connexion `config` ; celle qui donne accès au serveur à tous les utilisateurs qui connaissent l'adresse de votre phpMyAdmin.

Sécuriser malgré tout
Si vous tenez à conserver le mode de connexion "config", vous pouvez tout de même sécuriser l'accès à votre serveur de bases de données en utilisant les restrictions d'accès que propose votre serveur web (ex. : fichier .htaccess d'Apache).

Vous pouvez également configurer dès à présent phpMyAdmin pour qu'il soit en français. Recherchez cette ligne dans le fichier *config.inc.php* :

```
require('./libraries/select_lang.lib.php');
```

et modifiez-la comme suit :

```
require("french.inc.php3");
```

Une autre option intéressante de ce fichier de configuration est la possibilité de configurer l'accès pour des utilisateurs différents ayant des droits différents. Pour utiliser cette fonctionnalité de phpMyAdmin, il faut bien entendu que des utilisateurs autres que "`root`" aient été créés sur le serveur MySQL (voir plus loin *Prise en main*). Dans le fichier de configuration, vous devez trouver plusieurs fois cette suite de lignes :

```
$i++;
$cfgServers[$i]['host']          = '';
$cfgServers[$i]['port']          = '';
$cfgServers[$i]['socket']        = '';
$cfgServers[$i]['connect_type']  = 'tcp';
$cfgServers[$i]['controluser']   = '';
$cfgServers[$i]['controlpass']   = '';
$cfgServers[$i]['auth_type']     = 'config';
$cfgServers[$i]['user']          = 'root';
$cfgServers[$i]['password']      = '';
$cfgServers[$i]['only_db']       = '';
$cfgServers[$i]['verbose']       = '';
$cfgServers[$i]['bookmarkdb']    = '';
$cfgServers[$i]['bookmarktable'] = '';
$cfgServers[$i]['relation']      = '';
```

Ce sont des doublons de configuration d'accès prêts à l'emploi. Il suffit de les dupliquer et de les compléter pour chacun des utilisateurs requis. Pour un phpMyAdmin qui dispose déjà d'un utilisateur "`root`", si l'on veut ajouter l'utilisatrice "`Emma`", ayant le mot de passe "`youpi`", il faudra avoir la configuration suivante :

```
$cfgServers[1]['host']          = 'localhost';
$cfgServers[1]['port']          = '';
$cfgServers[1]['socket']        = '';
$cfgServers[1]['connect_type']  = 'tcp';
$cfgServers[1]['controluser']   = '';
$cfgServers[1]['controlpass']   = '';
$cfgServers[1]['auth_type']     = 'http';
$cfgServers[1]['user']          = 'root';
$cfgServers[1]['password']      = 'coucou';
$cfgServers[1]['only_db']       = '';
$cfgServers[1]['verbose']       = '';
$cfgServers[1]['bookmarkdb']    = '';
$cfgServers[1]['bookmarktable'] = '';
$cfgServers[1]['relation']      = '';

$cfgServers[2]['host']          = 'localhost';
$cfgServers[2]['port']          = '';
$cfgServers[2]['socket']        = '';
$cfgServers[2]['connect_type']  = 'tcp';
$cfgServers[2]['controluser']   = '';
$cfgServers[2]['controlpass']   = '';
$cfgServers[2]['auth_type']     = 'http';
$cfgServers[2]['user']          = 'emma';
$cfgServers[2]['password']      = 'youpi';
$cfgServers[2]['only_db']       = '';
```

```
$cfgServers[2]['verbose']       = '';
$cfgServers[2]['bookmarkdb']    = '';
$cfgServers[2]['bookmarktable'] = '';
$cfgServers[2]['relation']      = '';
```

La ligne suivante :

```
$cfgServers[2]['only_db']       = '';
```

permet ensuite de spécifier les bases qui doivent être listées pour cette utilisatrice en particulier. Cela ne remplace en rien les droits et privilèges attribués dans MySQL. Il est seulement question d'affichage.

Une fois que votre fichier de configuration est correctement renseigné, sauvegardez-le. Vous pouvez alors commencer à utiliser phpMyAdmin en vous rendant à l'adresse : **http://votrenomdedomaine/phpmyadmin/**.

Prise en main

phpMyAdmin vous permet de gérer au mieux vos bases MySQL. Nous allons passer en revue les fonctions les plus simples et les plus basiques.

Aspect général

Dans la partie gauche de la page, vous devez trouver, sous la mention *Accueil*, un menu déroulant des différentes bases de données présentes sur le système. La partie qui nous intéresse se trouve sur la partie droite de la page. On peut voir deux colonnes : une première qui propose des liens en rapport avec MySQL, et une seconde qui concerne phpMyAdmin.

Créer un utilisateur

Pour créer un utilisateur, rendez-vous sur le page principale de votre phpMyAdmin. Dans la partie droite de la page, sous *MySQL* cliquez sur le lien *Utilisateurs et privilèges*. Notez au passage que vous disposez également d'un lien vers la documentation en ligne. Sur la nouvelle page qui s'affiche, vous trouvez deux parties : en premier lieu, un tableau qui résume tous les utilisateurs déjà créés ainsi qu'un résumé de leurs privilèges et, en second lieu, une série d'options et de champs qui vont vous permettre de créer de nouveaux utilisateurs. Allez directement dans la partie *Ajouter un utilisateur* (voir fig. 21.2).

Vous pouvez tout d'abord sélectionner le serveur sur lequel vous désirez créer cet utilisateur, phpMyAdmin pouvant effectivement gérer plusieurs serveurs. La ligne suivante vous demande *Tout utilisateur* ou *Nom d'utilisateur* si vous souhaitez spécifier un nom d'utilisateur. Cela est d'ailleurs très largement préférable. En effet, vous pourriez créer un accès sur n'importe quel nom d'utilisateur sans donner de mot de passe, ce qui aurait pour effet d'ouvrir votre serveur de bases de données à n'importe qui...

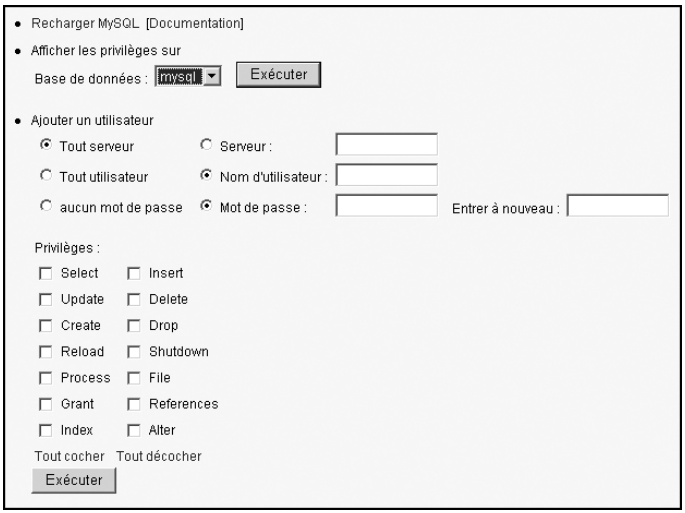


Figure 21.2 : *Ajouter un utilisateur dans phpMyAdmin*

Donnez donc un nom d'utilisateur dans le champ prévu à cet effet, et donnez deux fois de suite le mot de passe sur la ligne suivante. Il faut encore indiquer les privilèges qui seront attribués à ce nouvel utilisateur. Cochez chacune des cases correspondantes selon les privilèges que vous souhaitez allouer. Vous pouvez cliquer sur **Exécuter** pour que la création soit lancée.

Une fois ce nouvel utilisateur créé, vous pouvez l'ajouter à votre fichier de configuration de phpMyAdmin pour qu'il puisse se connecter au serveur, ou l'utiliser pour des applications web nécessitant une connexion au serveur de base.

Créer une base

Pour créer une base, cliquez sur *Accueil*, tout en haut à gauche de la page d'accueil de phpMyAdmin. Une fois la page rafraîchie, juste sous la petite barre *MySQL*, donnez le nom de table que vous voulez créer dans le champ prévu à cet effet, puis cliquez sur **Créer**.

Supprimer une base

Pour supprimer une base, choisissez la base dans le menu déroulant qui se situe juste sous la mention *Accueil*. La partie de droite s'actualise alors pour vous présenter un résumé de la table sélectionnée. Tout en bas de cette page se trouve un lien *Supprimer la base ...* Il suffit de cliquer sur ce lien.

Créer une table

Une fois que vous avez choisi une base, sur la partie droite de la page web, rendez-vous dans la section *Créer une nouvelle table sur la base ...* Donnez alors le nom de la table ainsi que le nombre de champs que vous voulez qu'elle comporte, puis cliquez sur *Exécuter*. La page se rafraîchit et vous présente alors une vue en tableau de la future table. Renseignez tous les paramètres nécessaires pour chaque champ de la table, puis validez en cliquant sur le bouton **Sauvegarder** situé en bas de la page.

Supprimer une table

Choisissez d'abord la base dans laquelle se trouve la table. Dans le menu situé à gauche de la page, cliquez sur le nom de la table que vous voulez supprimer. Dans la partie droite de la page, cliquez sur le lien *Supprimer* de la ligne correspondant à la table concernée.

Sauvegarder une table ou une base

Sauvegarder une base ou une table MySQL est une opération fort simple grâce à phpMyAdmin. Cliquez, dans la partie gauche de la page, sur une table ou une base. La partie droite se rafraîchit. Rendez-vous dans la section *Afficher le schéma* de la base (ou de la table). Si vous avez choisi une base, vous pourrez sélectionner les tables à sauvegarder grâce à une petite liste. Ensuite, décidez de sauvegarder la structure de la base, la structure et les données ou seulement les données. Choisissez ensuite les options qui vous intéressent parmi les différentes options offertes : ajouter des énoncés, insertions complètes ou étendues, etc. Cliquez sur **Transmettre**. Selon les configurations de PHP, il vous sera également possible de compacter le fichier de sauvegarde (zippé).

Restaurer une table ou une base

Le fichier généré par la sauvegarde peut, bien sûr, servir à recréer une table ou une base. En fait, le fichier de sauvegarde contient des requêtes SQL et des données variables selon les options choisies lors de la sauvegarde. Admettons que votre base portait le nom de "`Emma`" : il vous suffit de récréer une base "`Emma`" sur le nouveau serveur. Rendez-vous dans la section *Exécuter une ou des requêtes sur la base Emma*. Dans la partie *Emplacement du fichier texte* , cliquez sur le bouton **Parcourir**, donnez l'emplacement du fichier de sauvegarde puis cliquez sur **Exécuter**.

Autres

Il existe d'autres scripts, similaires à phpMyAdmin mais adaptés à d'autres serveurs de bases de données. Parmi eux nous trouvons :

- phpPgAdmin, destiné à administrer PostgreSQL et disponible à l'adresse **http://sourceforge.net/projects/phpPgAdmin/** ;
- phpOracleAdmin, destiné à administrer Oracle et disponible à l'adresse **http://phporacleadmin.org/**.

21.2. Création de sites

PHPNuke

Présentation

PHPNuke est un logiciel libre et gratuit que vous pourrez modifier et adapter à votre gré.

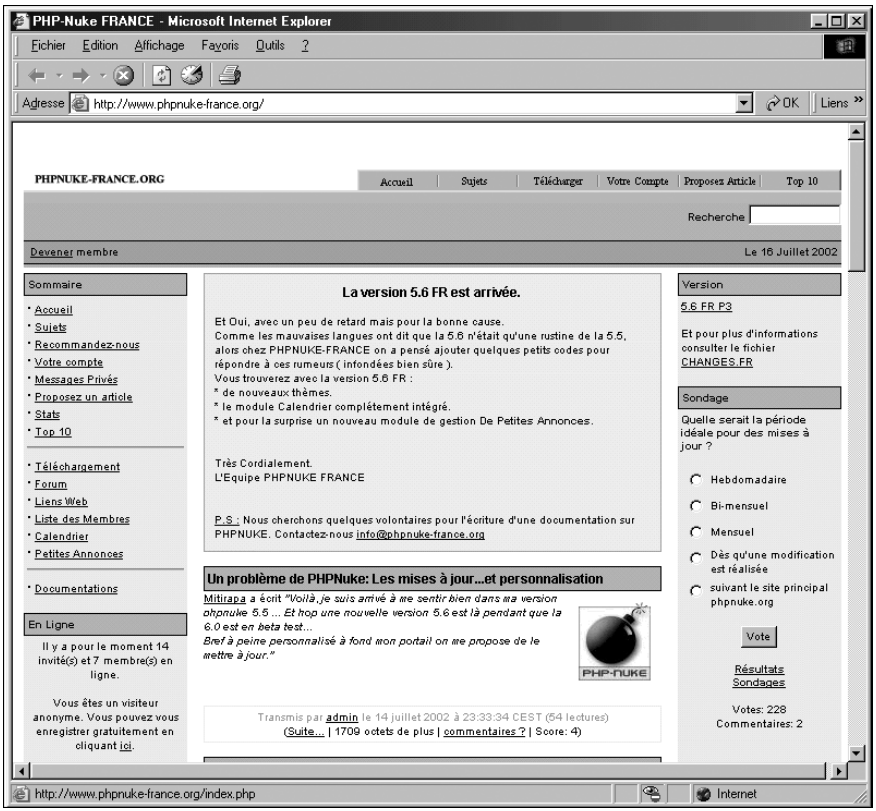


Figure 21.3 : *PHPNuke*

PHPNuke est un système qui gère complètement votre site web. Lorsque vous installez PHPNuke, vous obtenez un site qui a déjà une page d'accueil vous permettant de mettre en ligne des articles avec forums intégrés ; il gère l'inscription des utilisateurs, offre une zone de téléchargement modérée, etc. Vous n'avez pas à utiliser un éditeur de pages web pour travailler avec un site qui utilise PHPNuke. S'il est possible de reprocher à PHPNuke d'être une "usine à gaz", il reste tout de même une solution rapide et pratique pour mettre en place un site et diffuser des informations sans avoir à passer par les étapes fastidieuses de l'éditeur HTML et du développement en PHP.

Au quotidien, deux aspects de PHPNuke seront mis face à face : sa facilité de gestion et sa rigidité. Par exemple, la page d'accueil d'un site sous PHPNuke se présente sous forme de

blocs : blocs de menus, blocs des articles, etc. Si vous voulez faire passer un bloc de liens de droite à gauche, il vous suffit de cliquer sur de petites flèches dans l'espace d'administration du site. Si vous voulez mettre en ligne un article, vous faites un copier-coller dans un champ texte et le tour est joué. De même, pour changer tout l'aspect visuel du site, il n'y a qu'à choisir dans une liste pour que tout le look du site soit modifié. En revanche, vous risquez d'être rapidement bloqué par la rigidité de PHPNuke. Pour ne parler que d'un point relativement secondaire dans un site web : la page d'accueil reste désespérément figée. Vous ne pourrez pas sortir du modèle "deux petites colonnes et une grosse" sans mettre les mains dans le code. Cela risque d'ailleurs de faire s'effondrer le château de cartes, tout étant très imbriqué dans PHPNuke. À la longue, cela peut lasser.

Installation

Pour installer PHPNuke, allez tout d'abord créer un dossier "*phpnuke*" quelque part sur votre serveur web ou chez votre hébergeur. Décompactez l'archive PHPNuke disponible à l'adresse **http://www.phpnuke.org/** (mais également présente sur le CD-ROM de la présente Bible). Il faut décompacter le contenu du dossier "*/html*" de l'archive dans le dossier " *phpnuke*". Le reste se résume à une licence que vous devez déjà connaître par cœur (la GPL) et à des fichiers *README* (assez classiques, mais que les anglophones auront tout de même intérêt à lire pour être tenus au courant des évolutions de PHPNuke).

Après avoir décompacté tous les fichiers, rendez-vous dans phpMyAdmin. Pour que PHPNuke fonctionne, il faut qu'il puisse utiliser une base de données MySQL. Si vous êtes votre propre hébergeur, vous pouvez envisager de créer une table "nuke", sinon toute autre base devrait faire l'affaire.

Vous pouvez vous reporter à la section MySQL du chapitre sur les bases de données ou à l'annexe phpMyAdmin pour voir comment créer une telle base.

Il faut maintenant créer les tables, qui sont en fait les tiroirs dans lesquels PHPNuke va venir stocker ses informations. Rassurez-vous, les développeurs de PHPNuke ont pensé à vous. Dans le dossier qui vous a servi à décompacter PHPNuke, vous devriez trouver le fichier *nuke.sql*. C'est ce fichier qui contient toute la structure et la définition de PHPNuke utiles à votre base de données.

Si vous avez accès au client MySQL en ligne de commande (i.e. si vous installez PHPNuke sur votre propre machine), vous pouvez taper la commande `mysql mabase < nuke.sql` (si votre base s'appelle "mabase", vous aurez peut-être à spécifier un nom d'utilisateur et un mot de passe. Pour plus de détails, tapez alors `mysql -h`).

Sinon, vous pouvez utiliser phpMyAdmin. Dans la liste des bases présentes sur la partie gauche de la page web de phpMyAdmin, sélectionnez celle qui doit héberger les tables PHPNuke. La partie droite de la page doit se rafraîchir. Sur cette page, vous trouvez différentes options qui vont vous permettre d'effectuer des requêtes sur votre base. Un lien "*Emplacement du fichier texte*" vous offre de télécharger un fichier qui contient toute une série de requêtes. C'est cette solution que nous allons utiliser pour créer toutes les tables dont a besoin PHPNuke. Dans la page web, cliquez sur Parcourir, allez chercher le fichier *nuke.sql*, puis cliquez sur le bouton Exécuter pour que les informations contenues dans le fichier soient envoyées au serveur. Si jamais, comme cela a pu arriver, l'opération produisait une erreur, vous pouvez essayer de

rentrer les requêtes de création de tables une à une. Opération certes fastidieuse, mais nécessaire au bon fonctionnement de PHPNuke.

Une fois que le serveur a digéré toutes les requêtes, vous devez avoir une belle liste de tables dans votre base (sur le côté gauche de la page).

Une fois les tables créées, il faut configurer PHPNuke afin qu'il sache où aller chercher ces bases. À l'aide de votre éditeur de texte préféré, ouvrez le fichier *config.php* qui se situe à la racine de votre site. À l'intérieur de ce fichier, recherchez les lignes suivantes (en début de fichier) :

```
$dbhost = "localhost";
$dbuname = "toto";
$dbpass = "passtoto";
$dbname = "basetoto";
$dbtype = "MySQL";
```

Tableau 21.1 : Un passage commenté du fichier vous explique à quoi correspond chacune des lignes

Paramètre	Signification
`$dbhost`	Nom de l'hôte de la base MySQL. Généralement, laissez "localhost". Renseignez-vous auprès de votre hébergeur pour savoir quels sont les paramètres exacts.
`$dbuname`	Nom de l'utilisateur de la base MySQL.
`$dbpass`	Mot de passe de l'utilisateur.
`$dbname`	Nom de la base MySQL.

Une fois que vous avez correctement renseigné chacune de ces lignes, sauvegardez le fichier. Voilà, tout est dit. Votre site avec PHPNuke est accessible en ligne, chez votre hébergeur ou sur votre serveur local. Pour le consulter, rendez-vous directement à la racine du répertoire dans lequel vous avez installé PHPNuke, par exemple, http://www.votresite.com/test/. Si rien n'apparaît, ou si vous avez des erreurs MySQL, cela provient peut-être d'une erreur lors de l'édition du fichier config.php. Vérifiez bien les informations que vous avez données. Si tout est bon, vous devez alors voir la page d'accueil de votre site. Outre le message central, les colonnes de droite et de gauche sont déjà bien pleines. Vous pourrez réduire, configurer ou supprimer ces blocs latéraux depuis l'espace d'administration qui est accessible à une adresse du type : http://www.votresite.com/test/admin.php. Rendez-vous d'ailleurs à cette adresse pour créer votre compte d'administrateur.

Figure 21.4 : *Création d'un compte d'administrateur*

Il vous sera demandé un nom d'utilisateur, une adresse email, l'adresse du site de l'administrateur (vous pouvez donner l'adresse de votre site) ainsi qu'un mot de passe. Une fois toutes les informations données, cliquez sur le bouton Submit. Vous arrivez alors directement sur une page qui vous permet de vous connecter en tant qu'administrateur.

Figure 21.5 : *Connexion à l'administration*

En plus du nom d'utilisateur et du mot de passe que vous venez de définir, PHPNuke vous demander de saisir un code à cinq chiffres aléatore affiché à l'écran. Cela évite les tentatives de piratage par des robots qui testent des jeux de noms d'utilisateur et mot de passe.

SPIP

Présentation (simple, puissant, beau)

Figure 21.6 : *SPIP*

SPIP est un moteur de gestion de contenus. Il vous permet d'écrire des articles et de les mettre en ligne. Entendez par "article" tout contenu que vous jugerez bon de mettre à disposition des visiteurs de votre site. SPIP a été crée par l'équipe du site *Uzine*. Par rapport à PHPNuke, SPIP a quelques avantages. Il dispose notamment d'un système de cache qui évite de surcharger un serveur, alors que PHPNuke, lui, est lourd au possible. Signalons au passage que ce système de cache a pour nom "Gargantua"… voilà qui ravira tous les amateurs de littérature française. SPIP est aussi plus simple à mettre en place et à gérer. Étant plus jeune que PHPNuke, SPIP dispose pour le moment de moins de fonctionnalités. Gageons que ses créateurs sauront lui garder cette simplicité qui en fait un excellent outil. Soulignons toutefois que SPIP dispose de petits "plus" tout à fait séduisants. Par exemple, quand un auteur se crée un compte, il a la possibilité de donner sa clé publique PGP… SPIP est actuellement en train de s'orienter vers le travail collaboratif (système de forum interne à certains rédacteurs, messages privés, etc.). On reprochera tout de même à SPIP le manque de "une". En effet, la une d'un site géré par SPIP est en constant mouvement. À chaque fois que vous ajoutez un article, il vient prendre place sur la page d'accueil. Même si vous annoncez la périodicité de vos mises à jours, vos visiteurs seront privés du plaisir de découvrir vos anciennes pages d'accueil. Les articles sont bien archivés, mais la une, elle, n'a pas de mémoire.

Compatibilité PHP5
La version 1.7.2 de SPIP n'est pas encore compatible avec PHP5. Les développeurs de SPIP y travaillent en modifiant le code du moteur. Ainsi, SPIP devrait rapidement devenir compatible avec la version la plus récente de PHP. Pour plus d'informations : **www.spip.net/threadspip2014-7420.html**.

Installation (du travail de pro)

Grossièrement, le processus est le même que pour PHPNuke. Il va s'agir de mettre en place une interaction entre SPIP et la base de données MySQL. Créez un répertoire *spip* dans le répertoire racine de votre serveur web, ou bien là où vous jugerez utile d'installer SPIP. Dans ce répertoire *spip*, décompactez le fichier *spip1.7.2.zip* disponible en téléchargement à l'adresse www.uzine.net/spip/ (et que vous trouverez sur le CD-ROM de la Bible). Si vous décompactez les fichiers à la racine de votre serveur web ou de votre espace d'hébergement, tout sera mis en vrac à la racine et, avouons-le, ce n'est pas très propre comme installation. Une fois les fichiers installés correctement, lancez un navigateur web et rendez-vous à cette adresse : www.votresite.com/spip/ecrire/. Là, et c'est ce qui est merveilleux, l'Assistant d'installation de SPIP va vous prendre par la main. Une première page vous demande l'adresse de votre base de données, le nom d'utilisateur à utiliser (login de connexion) puis le mot de passe de cet utilisateur. Nous ne pouvons que vous conseiller d'assigner un mot de passe à l'utilisateur que vous allez créer. Une fois tous les champs correctement remplis, cliquez sur **suivant**. Si la connexion échoue, SPIP vous explique pourquoi, et vous invite gentiment à corriger l'erreur (qui, entre nous soit dit, doit être assez bénigne). Une fois la connexion avec la base établie, cliquez sur **suivant** pour passer à la création de la base de données qui sera utilisée par SPIP. SPIP vous propose alors plusieurs choix de bases de données. Vous pouvez utiliser une base de données qui vous a été attribuée par votre hébergeur (quand vous n'avez, par exemple, droit qu'à une seule base par serveur de bases de données), ou bien créer une nouvelle base spécialement pour SPIP. SPIP vous propose d'ailleurs d'en créer une qui portera son nom, ce qui n'est pas une mauvaise idée (c'est d'ailleurs ce que nous vous invitons à faire). Ne touchez à rien, et cliquez, vous vous en doutez, sur le bouton **suivant**. SPIP va alors créer les tables nécessaires à son bon fonctionnement au sein de la nouvelle base de données. Une fois qu'il a fini son travail, il vous en informe, et vu que vous vous en tirez parfaitement bien, cliquez une nouvelle fois sur **suivant**. Tout le cœur du système SPIP est maintenant installé. Il faut désormais créer un premier utilisateur, un chef suprême : vous. Donnez votre signature, votre adresse e-mail puis un pseudonyme et un mot de passe qui vous serviront à l'administration de votre site. Notez-les précieusement, vous en aurez besoin par la suite. Une fois tous les champs remplis, cliquez sur **suivant**. SPIP vous annonce que vous avez tout gagné, que l'installation et la configuration sont terminées. Vous allez pouvoir commencer à travailler avec SPIP. Cliquez, et ce pour la dernière fois, sur **suivant**. Une petite fenêtre d'identification apparaît. Donnez le pseudo et le mot de passe que vous avez utilisés lors de l'installation. Vous arrivez alors dans le monde merveilleux de SPIP.

Configuration

L'interface d'administration est un peu ardue à comprendre, mais, rassurez-vous, une aide complète est intégrée. Vous pouvez y accéder en cliquant sur le bouton rouge en haut à droite. Par la suite, vous pourrez accéder à cette page d'administration en vous rendant à cette adresse : http://www.votresite.com/spip/ecrire. Vos pseudo et mot de passe vous seront redemandés.

Comme vous pouvez le constater, en vous rendant sur la page d'accueil du site que vous avez créé avec SPIP (http://www.votresite.com/spip), vous manquez cruellement de contenu. Pour remédier à cela, rendez-vous dans l'espace d'administration (http://www.votresite.com/spip/ecrire). SPIP lui-même vous l'indique : avant de pouvoir écrire, il faut que vous créiez au moins une rubrique. Cliquez sur le lien prévu à cet effet : *Créer une nouvelle sous-rubrique*. Vous arrivez alors sur une page où il vous est demandé de donner un titre à cette nouvelle rubrique et, si vous le désirez, de lui adjoindre un descriptif. Une fois que vous êtes satisfait de votre rubrique, cliquez sur le bouton **valider** situé en bas de la page. Vous retournez alors dans votre espace d'administration. Vous pouvez constater que la nouvelle rubrique apparaît. De nouvelles icônes sont également présentes là où ne figurait que celle de la création d'une rubrique. Notez tout particulièrement l'icône *Écrire un nouvel article*. Cliquez dessus. Vous arrivez alors sur une page vous offrant de rédiger un article. Une fois votre article terminé, enregistrez-le dans la base. SPIP vous propose alors un aperçu de votre travail et vous donne la possibilité de changer son statut. Étant donné que vous êtes l'administrateur du site, vous pouvez mettre directement ce travail en ligne, ce que nous vous invitons d'ailleurs à faire. À l'aide du menu déroulant présent tout en haut de la page, choisissez l'option *Publié en ligne*, puis cliquez sur **Modifier**. Vous pouvez aller récolter le fruit de votre dur labeur sur la page d'accueil de votre site : http://www.votresite.com/spip/.

Autres

Il existe de nombreuses solutions destinées à faciliter la mise en place d'un site. En voici quelques autres :

- Templeet, disponible à l'adresse http://www.templeet.org/ ;
- AttilaPHP, disponible à l'adresse http://www.attila-php.net/ ;
- PHPForge, disponible à l'adresse http://membres.lycos.fr/phpforge/ ;
- postNuke (un clone de PHPNuke), disponible à l'adresse http://www.postnuke.com/.

21.3. Forums de discussion

PHPbb

Présentation (cyber-agora et café du commerce virtuel)

Plutôt que de paraphraser bêtement, donnons la parole au site web *PHPbb.biz*, le site français de référence en ce qui concerne PHPbb : "Phpbb est un puissant forum de discussion, convivial et agréable d'utilisation. Ses fonctionnalités et sa rapidité sont semblables voire supérieures aux différents forums payants "haut de gamme" du marché. En effet, PHPbb est libre et gratuit (sous licence GNU/GPL). Il dispose d'une interface facile à utiliser et d'une aide en ligne sous forme de FAQ ("Frequently Asked Questions" ou questions les plus fréquemment posées). Le panneau d'administration est complet et vous permettra une gestion aisée de votre plateforme de communication. Un support technique en français est disponible, il vous permettra de répondre à toutes vos questions".

Vous trouverez également sur ce site un guide d'installation. Gardez cela en tête, cela pourra vous être utile quand vous rechercherez des compléments d'information.

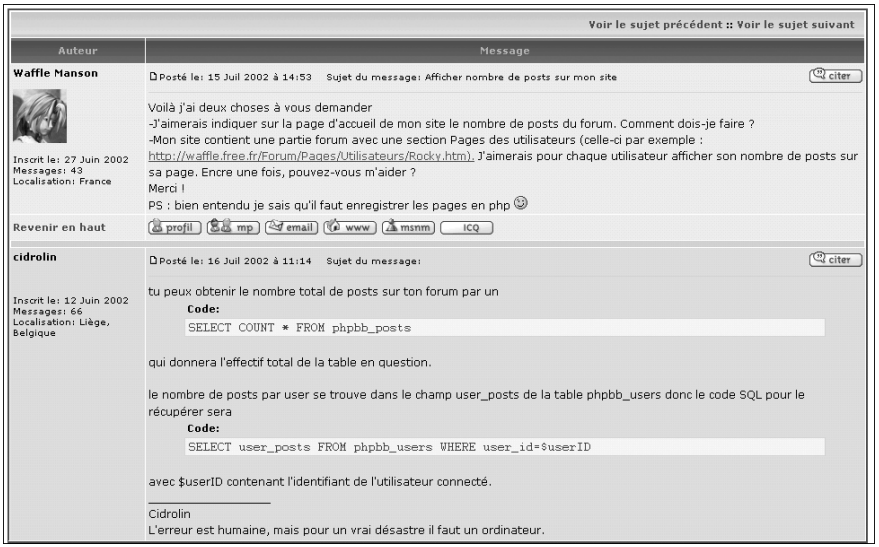


Figure 21.7 : *PHPbb*

Installation

Installez PHPbb en lui demandant de se décompacter directement dans un dossier que vous aurez créé pour lui. PHPbb créera lui-même un dossier *Phpbb*. Vous trouverez PHPbb sur le site http://www.phpbb.com/ (ou sur le site de la communauté francophone http://www.phpbb.biz) ainsi que sur le CD-ROM de la Bible. Lancez un navigateur et rendez-vous à l'adresse : http://www.votresite.com/phpBB/install.php. Une fois tous les champs correctement remplis, cliquez sur **next**. PHPbb va alors créer les tables nécessaires à son bon fonctionnement. Une page vous tient au courant de l'évolution des créations. Une fois que tout est fait, cliquez une nouvelle fois sur **next**. Il vous faut maintenant créer l'administrateur de votre système de forums. C'est lui qui aura droit de vie et de mort sur tout ce qui se passe dans les forums. Remplissez tous les champs en faisant particulièrement attention au nom d'utilisateur et au mot de passe. Une fois que vous avez rempli tous les champs, cliquez une nouvelle fois sur **next**. Vous arrivez alors sur la page de configuration générale du forum. Vous allez pouvoir, entre autres choses, choisir la langue du forum. Cliquez sur **next** pour enregistrer les paramètres. Voilà, le forum est configuré. PHPbb vous présente une page de félicitations et vous invite à aller dans l'espace d'administration des forums, ce qui, avouons-le, est plutôt une bonne idée. Cliquez sur le lien *Administration Area*. Vous arrivez alors sur une page assez laide, pleine de texte en anglais.

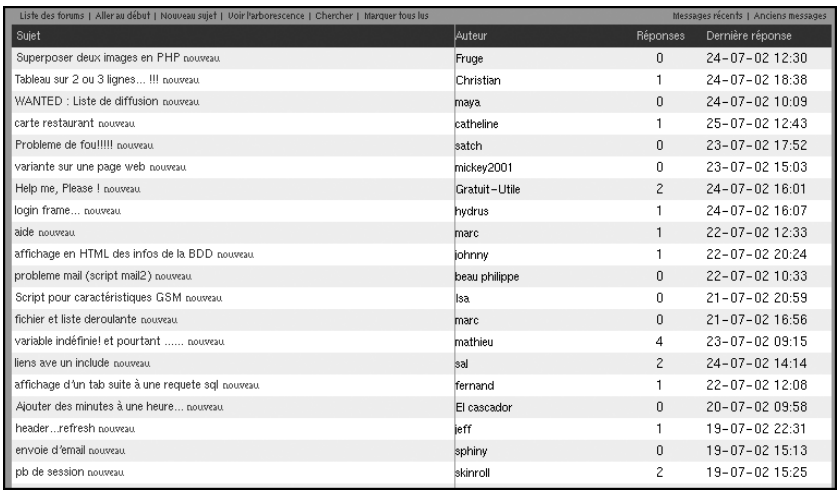
Configuration (la quête du Graal)

PHPbb vous prévient d'une faille de sécurité. Il refusera de fonctionner tant que vous n'aurez pas corrigé ce qui le tracasse. Pour les besoins de la configuration, le fichier *config.ini* était librement modifiable. Maintenant que la configuration est terminée et que vous avez créé un administrateur, PHPbb vous demande de changer les attributs du fichier. Pour cela, rendez-vous dans le répertoire "*phpBB*", sur votre serveur web ou chez votre hébergeur, et repérez le fichier *config.ini*. Dans votre explorateur de fichiers ou votre logiciel de FTP, modifiez les propriétés du fichier. Il faut qu'il soit en lecture seule. Vous pouvez retourner dans votre navigateur et rafraîchir la page sur laquelle vous avez laissé PHPbb. Vous voici désormais

dans l'espace d'administration de PHPbb. Si, de rage, vous aviez fermé votre navigateur, voici l'adresse : http://www.votresite.com/phpBB/admin/. Donnez les login et mot de passe de l'administrateur et cliquez sur **submit**. Vous arrivez alors dans le panneau d'administration de PHPbb. Il vous faut maintenant créer une nouvelle catégorie. Cliquez sur le lien *add a category*, donnez un nom de catégorie, "test" par exemple pour commencer, puis cliquez sur *create category*. Retournez sur la page d'administration en cliquant sur le lien *Panneau d'administration*. Vous allez maintenant devoir créer un forum en cliquant sur le lien *Add a forum.* Donnez un nom au forum, donnez une description, nommez des modérateurs (pour le moment, il n'y a que l'administrateur), choisissez une catégorie dans laquelle apparaîtra le forum (d'où l'intérêt d'avoir créé la catégorie avec le forum). Ne touchez pas les deux dernières options pour le moment ; vous pourrez les modifier et indiquer ainsi qui pourra écrire dans le forum et si le forum lui-même sera public ou privé. Cliquez sur *Create forum*. N'oubliez pas de cliquer sur le nom d'un modérateur avant de valider. Attention ! si vous oubliez un champ, PHPbb videra tous les autres champs lorsque vous reviendrez sur la page de configuration. Vous devrez repartir de zéro dans la définition du forum. Lorsque la création du forum a eu lieu, revenez sur la page d'accueil de votre site avec PHPbb. Vous allez toucher du doigt une petite difficulté de l'installation de PHPbb. Il vous faut cliquer sur le nom du forum que vous avez créé. Vous arrivez alors sur une nouvelle page : la page principale du forum. Vous allez devoir créer un sujet dans ce forum. Le lien se situe tout en haut à droite de la page web, c'est l'image *New Topic*. Cliquez dessus. Donnez alors un titre et une description pour ce sujet. Une fois cette opération effectuée, revenez sur la page principale du forum et cliquez sur le sujet que vous avez donné. Vous allez pouvoir entamer une discussion passionnante et enfiévrée.

21.4. Phorum : un moteur de forums

Phorum est un moteur de forums libre et gratuit. En plus d'être léger, il s'installe très facilement et très rapidement. Il dispose d'une aide à l'installation qui évite de passer par l'édition des fichiers de configuration.



Liste des forums | Aller au début | Nouveau sujet | Voir l'arborescence | Chercher | Marquer tous lus — Messages récents | Anciens messages

Sujet	Auteur	Réponses	Dernière réponse
Superposer deux images en PHP nouveau	Fruge	0	24-07-02 12:30
Tableau sur 2 ou 3 lignes... !!! nouveau	Christian	1	24-07-02 18:38
WANTED : Liste de diffusion nouveau	maya	0	24-07-02 10:09
carte restaurant nouveau	catheline	1	25-07-02 12:43
Probleme de fou!!!!! nouveau	satch	0	23-07-02 17:52
variante sur une page web nouveau	mickey2001	0	23-07-02 15:03
Help me, Please ! nouveau	Gratuit-Utile	2	24-07-02 16:01
login frame... nouveau	hydrus	1	24-07-02 16:07
aide nouveau	marc	1	22-07-02 12:33
affichage en HTML des infos de la BDD nouveau	johnny	1	22-07-02 20:24
probleme mail (script mail2) nouveau	beau philippe	0	22-07-02 10:33
Script pour caractéristiques GSM nouveau	lsa	0	21-07-02 20:59
fichier et liste deroulante nouveau	marc	0	21-07-02 16:56
variable indéfinie! et pourtant ..... nouveau	mathieu	4	23-07-02 09:15
liens ave un include nouveau	sal	2	24-07-02 14:14
affichage d'un tab suite à une requete sql nouveau	fernand	1	22-07-02 12:08
Ajouter des minutes à une heure... nouveau	El cascador	0	20-07-02 09:58
header...refresh nouveau	ieff	1	19-07-02 22:31
envoie d'email nouveau	sphiny	0	19-07-02 15:13
pb de session nouveau	skinroll	2	19-07-02 15:25

Figure 21.8 : *Phorum*

Phorum fonctionne avec MySQL ou PostgreSQL.

Téléchargez tout d'abord l'archive contenant Phorum directement sur le site de Phorum à l'adresse http://www.phorum.org, ou bien utilisez celle présente sur le CD-ROM de la Bible. N'oubliez pas de télécharger, en plus, le fichier de francisation *french.php* (fichier également disponible sur le CD-ROM).

Avant de vous lancer dans la procédure d'installation, il faut que vous créiez une nouvelle base sur votre serveur de bases de données. Elle sera utilisée par Phorum et vous devrez donner son nom lors de la procédure d'installation.

Pour commencez : décompactez l'archive de Phorum quelque part sur votre site web.

Une fois les fichiers copiés dans l'arborescence de votre site web, que ce soit en local ou chez un hébergeur, rendez-vous dans l'administration en ligne de Phorum pour paramétrer les forums. L'"admin" est accessible via l'URL : http://votresite/votredossierphorum/admin/index.php/. Choisissez une langue à utiliser pour l'installation puis cliquez sur le bouton **Next step**. Grâce au menu déroulant, choisissez le serveur de bases de données avec lequel devra travailler Phorum, puis cliquez sur le bouton **Submit**. Vous arrivez ensuite sur une page qui vous demande les paramètres de connexion à cette base.

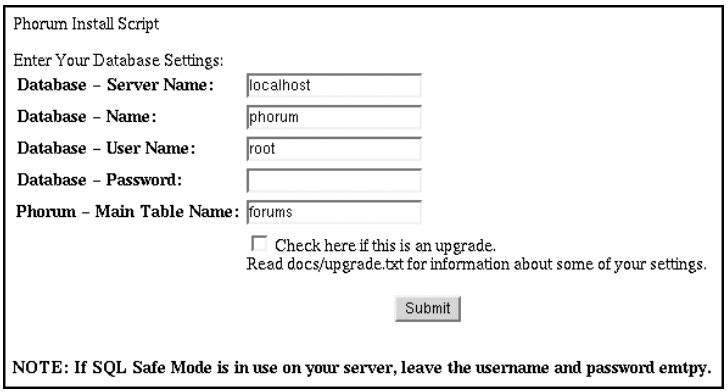


Figure 21.9 : *Configuration des paramètres de connexion*

Vous devrez donner le nom du serveur de bases de données, le nom de la base à utiliser (n'oubliez pas d'en créer une pour l'occasion), ainsi qu'un nom d'utilisateur et un mot de passe pour se connecter à cette base. Il faudra également spécifier le nom de la table qui sera utilisée par Phorum. Par défaut, le nom *forums* est donné automatiquement. Une fois tous les champs renseignés, cliquez sur **Submit**. Maintenant que la connexion à la base s'est effectuée et que les nouvelles tables sont ajoutées, Phorum vous invite à créer un administrateur. Donnez un nom d'utilisateur et un mot de passe dans les champs prévus à cet effet, puis cliquez une nouvelle fois sur **Submit**. Il ne reste alors plus qu'à donner l'adresse à laquelle le forum sera accessible et l'adresse e-mail de l'administrateur du forum.

Votre Phorum est maintenant installé. Mais, comme vous pouvez vous en rendre compte en allant sur sa page d'accueil, il est désespérément vide. Il faut en effet que vous amorciez la pompe en créant un premier forum.

Rendez-vous dans l'admin de votre forum (http://votresite/dossierphorum/admin/index.php).

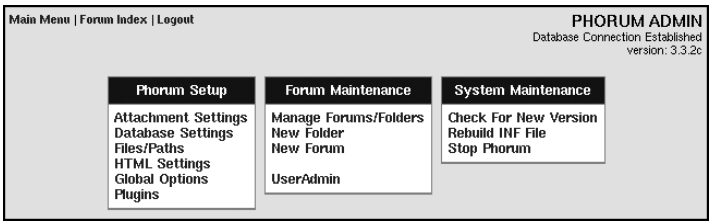


Figure 21.10 : *Page d'administration de Phorum*

Depuis l'admin, dans le bloc central *Forum Maintenance*, cliquez sur le lien *New Forum*. Vous arrivez alors sur une page qui permet la création d'un forum. Parmi toutes les options proposées, deux seulement sont obligatoires : le nom du nouveau forum et la table qu'il utilisera dans la base de données associée à votre Phorum. Une fois le forum créé, vous pouvez déjà l'utiliser. Vous n'êtes pas obligé de présenter vos forums directement les uns à la suite des autres. Vous avez la possibilité de créer des dossiers (folders) dans lesquels il est possible de regrouper des forums. Il suffit, pour cela, depuis l'admin, de créer un nouveau folder. Ensuite, lors de la création d'un nouveau forum, vous pourrez, à l'aide d'un menu déroulant, lui associer un folder particulier. Cette option est particulièrement pratique quand un site commence à avoir un certain nombre de forums...

N'oubliez pas de franciser l'interface utilisateur de Phorum. Vous n'avez qu'à copier le fichier *francais-3.3.2c.php* dans le dossier *lang* de l'arborescence de votre installation de Phorum. Une fois le fichier copié, rendez-vous dans la page d'administration.

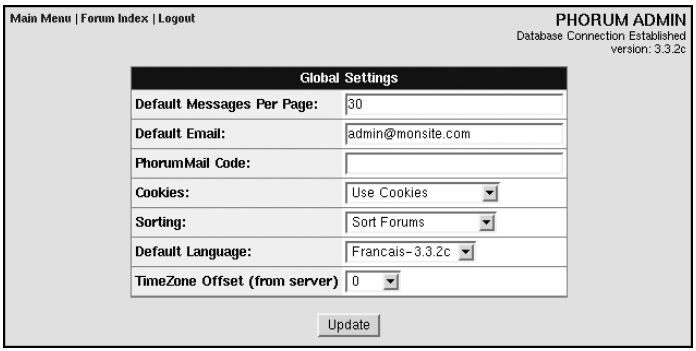


Figure 21.11 : *Sélection de la langue*

Cliquez sur le lien *Global Options*. À la ligne *Default Language*, dans le menu déroulant, vous pourrez alors choisir le français.

Autres

Il existe évidemment d'autres forums comme :

- NeoBoard, disponible à l'adresse **http://www.neoboard.net:8080/NeoBoard/** ;
- electrifiedForum (eF), disponible à l'adresse **http://www.electrifiedpenguin.com/mainindex.php** ;
- bestweb Forum, disponible à l'adresse **http://www.bestweb.ca/**.

Entre portail et forum :

- myWBBPortal, disponible à l'adresse http://www.wbbhacks.com/.

21.5. Annuaires de liens

Netref

Netref vous permet en quelques minutes d'installer un annuaire de liens sur votre site. Netref est une application écrite en PHP fonctionnant avec des bases de données MySQL, et qui se configure très facilement. Du même type que des annuaires tels que Yahoo ! par exemple, il vous permettra de gérer différentes catégories et sous-catégories, des suggestions de liens faites par les internautes, un classement des liens les plus visités. Son interface d'administration est claire, et vous permet, entre autres, de vérifier les liens soumis, de visualiser l'interface de votre annuaire, etc. Le programme est disponible en cinq langues : français, anglais, italien, suédois ou hollandais.

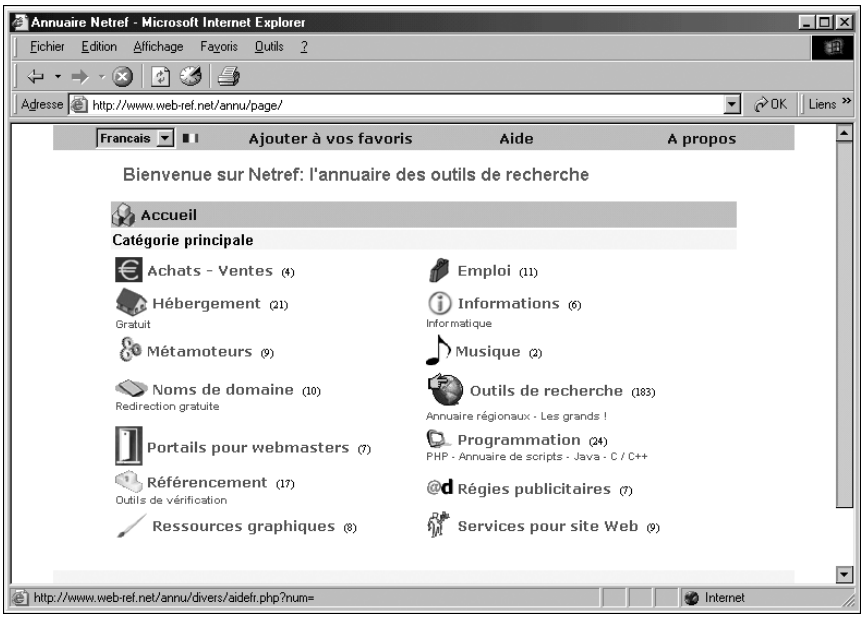


Figure 21.12 : *Une base de liens gérée avec Netref*

Tout d'abord, téléchargez le fichier *netref.zip* à l'adresse http://www.web-ref.net/annu/inscr/index.php. Dans l'arborescence de votre site, créez un répertoire *annuaire* à la racine de votre site, puis décompactez le fichier dans le répertoire *annuaire*. Vous devez ensuite paramétrer le fichier *option.php*.

Indiquez les paramètres de connexion à votre base de données.

```
$host="localhost";
```

Indiquez ici le nom d'hôte du serveur de bases de données.

```
$user="username";
```

Indiquez ici votre nom d'utilisateur utilisé pour votre connexion à la base de données.

```
$pass="password";
```

Indiquez ici votre mot de passe utilisé pour votre connexion à la base de données.

```
$bdd="basename";
```

Indiquez ici le nom de la base de données à utiliser.

Passez ensuite à l'administrateur de l'annuaire :

```
$psadmin[0]="sylgil";
```

Donnez ici le nom d'utilisateur qui doit être utilisé pour l'administrateur.

```
$passadmin[0]="alfred";
```

Donnez ici le nom d'utilisateur qui sera requis pour l'administrateur.

Enfin, publiez le répertoire *annuaire* sur votre espace d'hébergement, et rendez-vous sur la page **www.votresite.net/annuaire/admin/**. Entrez alors vos identifiants pour accéder au menu de l'administration, puis cliquez sur *Créer les tables sql*. Voilà, votre annuaire est prêt. Vous pouvez créer les catégories et sous-catégories à partir du menu d'administration.

Autres

Quelques autres annuaires de liens :

- phpMyAnnu, disponible à l'adresse **http://www.creation-de-site.net/** ;
- HitWeb, disponible à l'adresse **http://www.freesoftware.fsf.org/hitweb/** ;
- PHPMyLinks, disponible à l'adresse **http://rhenriot.free.fr/phpmylinks/**.

21.6. Solutions de travail collaboratif

Moregroupware

Moregroupware est un outil de groupware (travail collaboratif) très complet. Il vous permettra de gérer votre groupe de travail en toute simplicité. Vous pourrez créer différents groupes et utilisateurs, et leur attribuer des droits divers. Différents modules sont à votre disposition : un calendrier pour gérer plusieurs plannings, un répertoire, un module d'informations générales, un webmail, etc. Il peut se mettre en relation avec différents types de bases de données (MySQL, PostgreSQL, Oracle...). Il supporte plusieurs langues, dont le français.

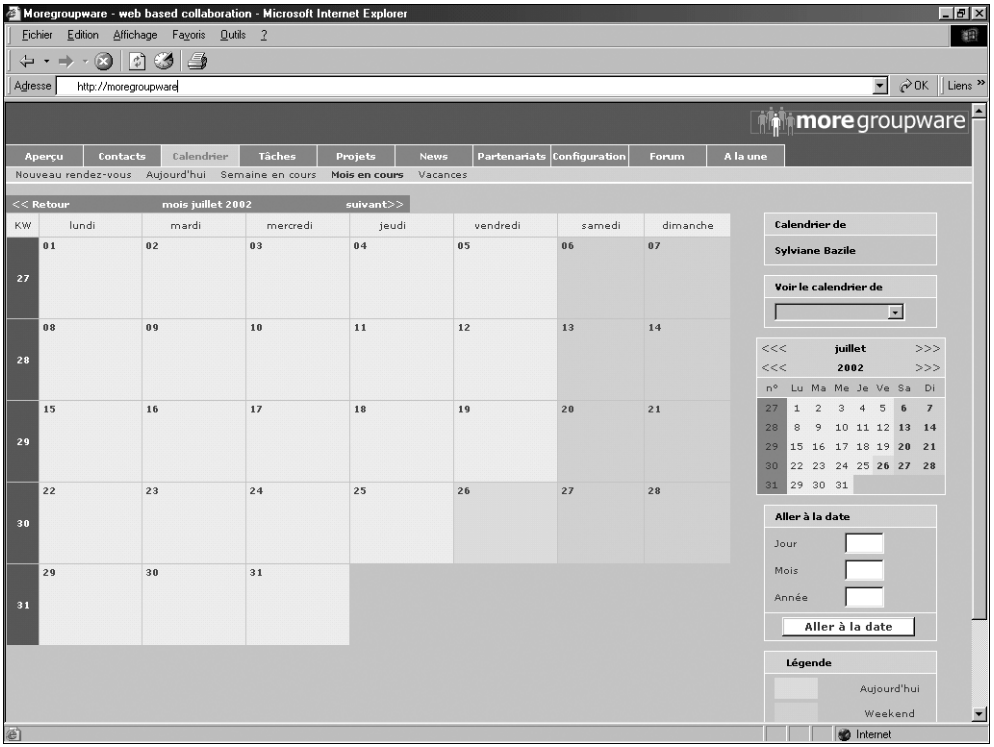


Figure 21.13 : *Moregroupware*

ATTENTION

Compatibilité PHP5

La version 0.7.2 de Moregroupware n'est pas encore compatible avec PHP5. Les développeurs y travaillent en modifiant le code du moteur. Ainsi, Moregroupware devrait rapidement devenir compatible avec la version la plus récente de PHP.

Installation

Avant tout, assurez-vous que votre *php.ini* est configuré avec la fonction `magic_quotes` activée. Téléchargez Moregroupware à l'adresse **http://www.moregroupware.org/download.php**, ou utilisez le fichier disponible sur le CD-ROM de la Bible. Décompactez le fichier à la racine de votre site, puis rendez-vous à l'adresse **http://www.monsite.com/moregroupware/**. Une page s'affichera vous demandant de configurer votre groupware. Cliquez alors sur le lien *Sélectionnez le langage* et entrez les données de connexion à votre base de données. Choisissez alors les modules que vous souhaitez installer puis validez. Un message de confirmation de l'installation apparaît. Rendez-vous alors à l'adresse qui vous sera indiquée en rouge pour administrer votre groupware. Vous devrez utiliser l'identifiant et le mot de passe par défaut : login `admin` et mot de passe `admin`.

Principales fonctions

Admin

Depuis l'interface de l'administrateur, vous pouvez gérer les utilisateurs, les groupes d'utilisateurs, les projets, etc. Rendez-vous tout d'abord sur l'interface d'administration pour changer le mot de passe administrateur et créez les utilisateurs. Entrez dans Moregroupware avec les mots de passe par défaut. Dans la partie *Admin*, puis dans *Gestionnaire d'utilisateurs*, éditez la fiche de l'utilisateur `admin`. Complétez-la, choisissez le mot de passe et le langage appropriés, puis validez. La procédure de création des autres utilisateurs est similaire. Vous pouvez ensuite attribuer les droits et les accès aux modules pour chacun des utilisateurs dans le gestionnaire des droits, et répartir les membres dans différents groupes.

Calendrier

Dans ce module, chaque utilisateur pourra gérer son planning, visualiser et/ou modifier celui des autres membres. Vous pourrez créer des événements ponctuels ou répétés.

Contacts

Cette rubrique vous permet de gérer des contacts aussi bien individuels que par groupe, par entreprise ou par fonction. Elle contient un moteur de recherche.

Aperçu

L'aperçu regroupe les événements vous concernant qui ont été planifiés : vos rendez-vous, les tâches que vous devez effectuer, etc.

Projets

Administrez les projets du groupe de travail : créez des projets, déterminez les membres qui interviendront dessus, créez des fiches de travail pour suivre l'évolution de chacun et obtenir un aperçu général de l'avancement du projet.

Configuration

Modifiez vos données personnelles, configurez votre webmail et déterminez vos préférences quant à l'affichage des diverses rubriques.

Tâches

Gérez les tâches concernant les différents projets : vos tâches personnelles, celles qui vous ont été déléguées, celles que vous avez déléguées. Vous pourrez suivre leur progression.

Webmail

Envoyez et recevez vos e-mails. Attention, n'oubliez pas de bien indiquer votre adresse dans le champ *adresse de retour* lors de la configuration, sans quoi vos destinataires ne pourront pas répondre automatiquement à vos e-mails.

Autres

Quelques solutions supplémentaires :

- PHPProjekt, disponible à l'adresse http://www.phpprojekt.com/ ;
- Tutos, disponible à l'adresse http://www.tutos.org/ ;
- Twig, disponible à l'adresse http://twig.screwdriver.net/.

Et il en existe bien d'autres…

21.7. Graphiques

JPGraph

Présentation

`JPGraph` est une bibliothèque en PHP qui permet de créer simplement et rapidement des graphiques en tous genres à partir de diverses sources. Avant d'aller plus loin, rendons grâce à Johann Persson, créateur de cet outil qu'il a bien voulu mettre librement à la disposition de tous les utilisateurs de PHP.

La génération dynamique de graphiques à partir de données statiques ou de variables peut constituer un réel "plus" pour un site web. Cela offre une vision synthétique de bien des sujets, tout en agrémentant des démonstrations qui, sinon, seraient bien austères. Les applications pratiques d'un générateur de graphiques, tableaux et autres camemberts n'ont plus à faire leurs preuves ; en témoigne le large succès d'un logiciel comme MS Excel. Toutes ces solutions sont maintenant à la portée de votre site web grâce à `JPGraph`.

`JPGraph` est une librairie en PHP orientée objet. Toutes les informations nécessaires à la réalisation d'un graphique sont regroupées dans un seul fichier PHP.

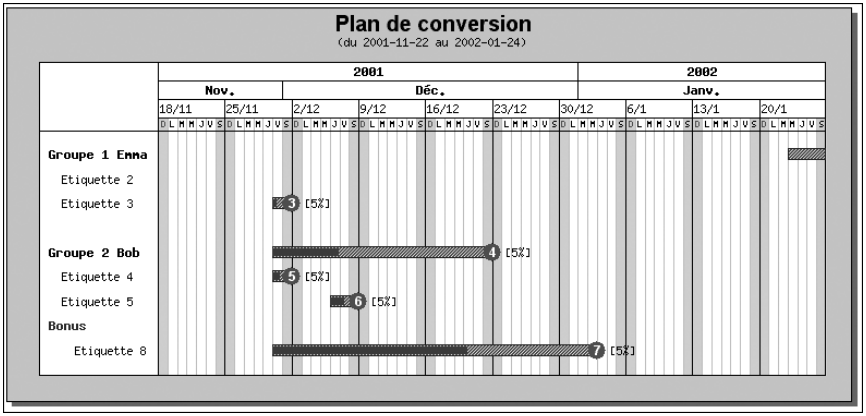


Figure 21.14 : *Un graphique généré à la volée par JPGraph*

Avant de vous lancer dans JPGraph, il faut vous assurer que vous disposez bien des outils nécessaires : un PHP version 4.02 ou ultérieure ainsi que le support de la librairie GD 1.x. Pour savoir si votre hébergeur répond aux conditions, faites un simple phpinfo() et vous serez vite fixé.


RENVOI

Vous pouvez consulter le chapitre "Les images et les animations Flash"

La génération d'un graphique ne demande que peu de lignes d'un code qui sera vite produit. Avant cela, il faut tout de même passer par l'installation et la configuration de JPGraph.


ATTENTION

Compatibilité PHP5
La version 1.16 de JPGraph ne supporte pas encore PHP5. Il vous faudra conserver une version antérieure de PHP (4.x) pour utiliser JPGraph ou bien attendre que la compatibilité avec PHP5 soit annoncée sur le site officiel du projet **www.aditus.nu/jpgraph**.

Installation

Installez directement JPGraph depuis le fichier zippé présent sur le CD-ROM, ou téléchargez une version plus récente en ligne (s'il en existe une) à l'adresse **http://www.aditus.nu/jpgraph/**. Décompactez les fichiers dans un dossier spécialement prévu à cet effet dans l'arborescence de votre serveur web. Le fichier zippé de JPGraph crée un dossier *jpgraph-1.16* au sein du répertoire dans lequel il s'installe (vous pouvez le renommer dossier). À la racine du dossier *JPGraph*, vous trouverez la licence de distribution du logiciel ainsi qu'un fichier *README* contenant des instructions d'installation (mais évidemment en anglais). Se trouve également dans le répertoire *JPGraph* un dossier *src*.

Les dossiers *Exemples* et *utils* contiennent, comme leur nom l'indique, des exemples d'utilisation de JPGraph ainsi que divers utilitaires pour aller plus loin dans l'utilisation de JPGraph.

Pour que vous puissiez travailler avec JPGraph, éditez le fichier *jpgraph.php* situé dans le dossier *src*. Vous avez globalement peu de paramètres à renseigner. Recherchez la ligne suivante :

```
DEFINE("CACHE_DIR","/tmp/jpgraph_cache/");
```

Vous devez donner un dossier temporaire qui sera utilisé par JPGraph. Faites bien attention qu'Apache puisse y écrire. Renseignez ensuite cette ligne :

```
DEFINE("TTF_DIR","/usr/local/fonts/ttf/");
```

Donnez là le chemin qui pointe vers les polices TrueType installées sur votre système (traditionnellement *C:\WINDOWS\FONTS* sous Windows).

Une fois ces deux lignes complétées, vous pouvez appeler directement le fichier *testsuit_jpgraph.php* depuis un navigateur web, en ayant pris soin de vous assurer que votre serveur web fonctionnait bien. Vous verrez alors toutes les potentialités de JPGraph.


REMARQUE

En cas de problème, appelez la police
Certains des exemples donnés par `JPGraph` *peuvent manquer à l'appel et ne pas fonctionner correctement. Dans la majeure partie des cas, cela vient d'une police utilisée qui n'est pas présente sur votre système. Deux solutions s'offrent à vous : modifier le fichier d'exemple pour spécifier une autre police ou bien installer celle qui manque à l'appel.*

Utilisation

`JPGraph` étant orientée objet, la création d'un graphique se résume à peu de choses. Voici un exemple de script :

```
<?php
include("local/jpgraph.php");
```

Vous devez faire un appel à *jpgraph.php* dans chacun des fichiers de création d'un graphique. À la place de "local", donnez le chemin vers le fichier *jpgraph.php* sur votre machine.

Dans notre cas, nous souhaitons créer un graphique en toile d'araignée ; nous devons donc également inclure le script *jpgraph_spider.php*.

```
include("local/jpgraph_spider.php");
```

Vous donnez ici le fichier de description du graphique que vous voulez créer. Vous trouverez dans le fichier *README* présent à la racine du dossier `JPGraph` une liste des fonctions utilisables. Pour savoir comment utiliser ces fonctions, les différents exemples inclus dans `JPGraph` pourront vous servir. Consultez également l'aide pour obtenir d'autres informations.

Les données doivent généralement est données via un tableau.

```
$data = array(30, 40, 50, 60, 70);
```

À chaque type de graphique correspond un objet (ici *SpiderGraph*) auquel est communiquée la taille de l'image.

```
$graph = new SpiderGraph(250,200,"auto");
$plot = new SpiderPlot($data);
```

Vous pouvez ensuite lancer la création du graphique avec les valeurs que vous avez affectées à la variable `$data`.

```
$graph->Add($plot);
$graph->Stroke();
?>
```

Pour utiliser une image générée par la librairie `JPGraph`, appelez celle-ci directement dans une balise `img`, comme ceci :

```
<img src="graphique1.php" border=0 align=center width=300 height=200>
```

Chapitre 22

Annexe B : les en-têtes HTTP et les variables externes

Vous trouverez dans le tableau suivant une liste des en-têtes HTTP les plus courants, qu'ils soient émis par le client à destination du serveur (C->S) ou par le serveur à destination du client (S->C).

Tableau 22.1 : En-têtes HTTP les plus courants

En-tête	Sens	Signification, exemples de valeurs, variable externe associée
`Accept`	C->S	Type de contenu que le navigateur est susceptible d'accepter (text/html, audio/x-aiff, image/jpeg, etc.) Il s'agit d'une chaîne de caractères construite par la concaténation des différents MIME-type supportés, séparés par une virgule. Une série de types peut être pondérée par un coefficient (indiquant une notion de préférence) ; la chaîne est alors suivie d'un point-virgule puis de 'q=' suivi du coefficient (<1).
	Exemple : `text/xml,text/html;q=0.9,text/plain;q=0.8, image/jpeg,q=0.2`	
	Cette valeur est accessible depuis `$_SERVER["HTTP_ACCEPT"]`.	
`Accept-Charset`	C->S	Le type de caractère que le navigateur attend en réponse (ex. : ISO-8859-1). Il s'agit d'une chaîne de caractères construite par la concaténation des différents encodages supportés, séparés par une virgule. Une série de types peut être pondérée par un coefficient (indiquant une notion de préférence) ; la chaîne est alors suivie d'un point-virgule puis de 'q=' suivi du coefficient (<1).
	Exemple : `ISO-8859-1, utf-8;q=0.66,*;q=0.33`	
	Cette valeur est accessible depuis `$_SERVER["HTTP_ACCEPT_CHARSET"]`.	
`Accept-Encoding`	C->S	Le type de compression que le navigateur est capable de supporter en réponse. Il s'agit d'une chaîne de caractères construite par la concaténation des différentes compressions supportées, séparées par une virgule. Une série de types peut être pondérée par un coefficient (indiquant une notion de préférence) ; la chaîne est alors suivie d'un point-virgule puis de 'q=' suivi du coefficient (<1).
	Exemple : `Gzip, deflate, compress;q=0.9`	
	Cette valeur est accessible depuis `$_SERVER["HTTP_ACCEPT_ENCODING"]`.	

En-tête	Sens	Signification, exemples de valeurs, variable externe associée
`Accept-Language`	C->S	La langue préférée de l'utilisateur. Il s'agit d'une chaîne de caractères construite par la concaténation des différents codes de langue supportés, séparés par une virgule. Une série de types peut être pondérée par un coefficient (indiquant une notion de préférence) ; la chaîne est alors suivie d'un point-virgule puis de '`q=`' suivi du coefficient (<1).
	Exemple : `fr, en;q=0.50`	
	Cette valeur est accessible depuis `$_SERVER["HTTP_ACCEPT_LANGUAGE"]`.	
`Referer`	C->S	Adresse de la page depuis laquelle la requête a été effectuée
	Exemple : `http://www.unannuairedelien.com/`	
	Cette valeur est accessible depuis `$_SERVER["HTTP_REFERER"]`.	
`User-Agent`	C->S	Chaîne d'identification du client.
	Exemple : `Mozilla/5.001 (windows; U; NT4.0; en-us) Gecko/25250101`	
	Cette valeur est accessible depuis `$_SERVER["HTTP_USER_AGENT"]`.	
`Host`	C->S	Nom de la machine cliente.
En-têtes de gestion du cache		
`Cache-Control`	S->C	Paramétrage du cache.
	Exemple : `max-age=60` (durée de vie du document limitée à 60 secondes).	
`Expires`	S->C	Limite de validité du document.
	Exemple : `Sat, 27 Apr 2002 16:00:53 GMT`	
`Pragma`	S->C	Gestion du cache.
	Exemple : `no-cache` (le document ne doit pas être mis en cache).	
Déclaration du contenu du document		
`Content-Type`	S->C	MIME-type du document émis par le serveur.
	Exemple : `text/html`	

En-tête	Sens	Signification, exemples de valeurs, variable externe associée
`Content-Encoding`	S->C	Type de compression (ou plus généralement codage) utilisé pour le document émis par le serveur.

	Exemple :
	`Compress`
	`Gzip`
	`Deflate`	

`Content-Length`	S->C	Taille (en octets) du document émis par le serveur.
`Content-Language`	S->C	Langue utilisée dans le document.
Redirection		
`Location`	S->C	Indique au client de se tourner vers une autre adresse.
	Exemple : `http://www.autredomaine.com`	
Informations serveur		
`Server`	S->C	Identifiant du serveur.
	Exemple : `Apache/1.3.26 (Unix) PHP/4.2.2`	
`Date`	S->C	Date et heure du serveur.
	Exemple : `Sat, 27 Apr 2002 15:59:53 GMT`	
En-têtes de mails		
`From`	Spécifie l'adresse de l'émetteur du mail.	
`To`	Spécifie les adresses des destinataires du mail.	
`Cc`	Spécifie les adresses des personnes mises en copie du mail.	
`Bcc`	Spécifie les adresses des personnes mises en copie cachée du mail.	
`Reply-To`	Spécifie l'adresse à laquelle le destinataire du mail doit répondre.	
`Subject`	Titre du mail.	
`MIME-Version`	Numéro de version de la norme MIME utilisée.	

Chapitre 23

Annexe C : les erreurs HTTP

Tableau 23.1 : Les principaux codes d'erreur que peut retourner un serveur web

Code	Signification
100 - 199	Les informations sont retournées.
200 - 299	La requête a été traitée avec succès.
200	Aucun incident à signaler.
204	Le "document" retourné est vide.
300 - 399	Demande de redirection.
300	Le serveur a besoin de plus d'informations.
301	Le document demandé a été déplacé définitivement.
302	Le document demandé a été déplacé temporairement (les requêtes suivantes ne nécessiteront pas forcément de redirection).
303	Le document doit être demandé à une autre adresse.
400 - 499	La requête est incomplète.
400	La requête n'est pas valide.
401	L'authentification a échoué. Si ce n'est pas le résultat attendu, sous Apache, vérifiez que les fichiers *.htaccess* et *.htpasswd* (le nom peut être différent) contiennent bien les nom et mot de passe que vous saisissez.
403	Accès interdit.
404	La page demandée au serveur n'a pu être trouvée.
405	La méthode utilisée est refusée par le serveur.
406	Authentification proxy exigée.
407	Authentification proxy exigée.
408	Délai d'attente de la requête du client dépassé.
411	Le serveur a besoin de la longueur de la requête.
413	La requête est trop longue.
500 - 599	Indique une erreur du serveur HTTP.
500	Erreur interne du serveur. Assurez-vous de ne pas avoir créé une boucle comme, par exemple, une série infinie de redirections (ce qui peut arriver lors des traitements automatiques des erreurs).
504	Délai d'attente dépassé.

Index des fonctions et méthodes

!

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

Index

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

Z

Composé en France par Jouve
11, bd de Sébastopol - 75001 Paris1349